The Transactor

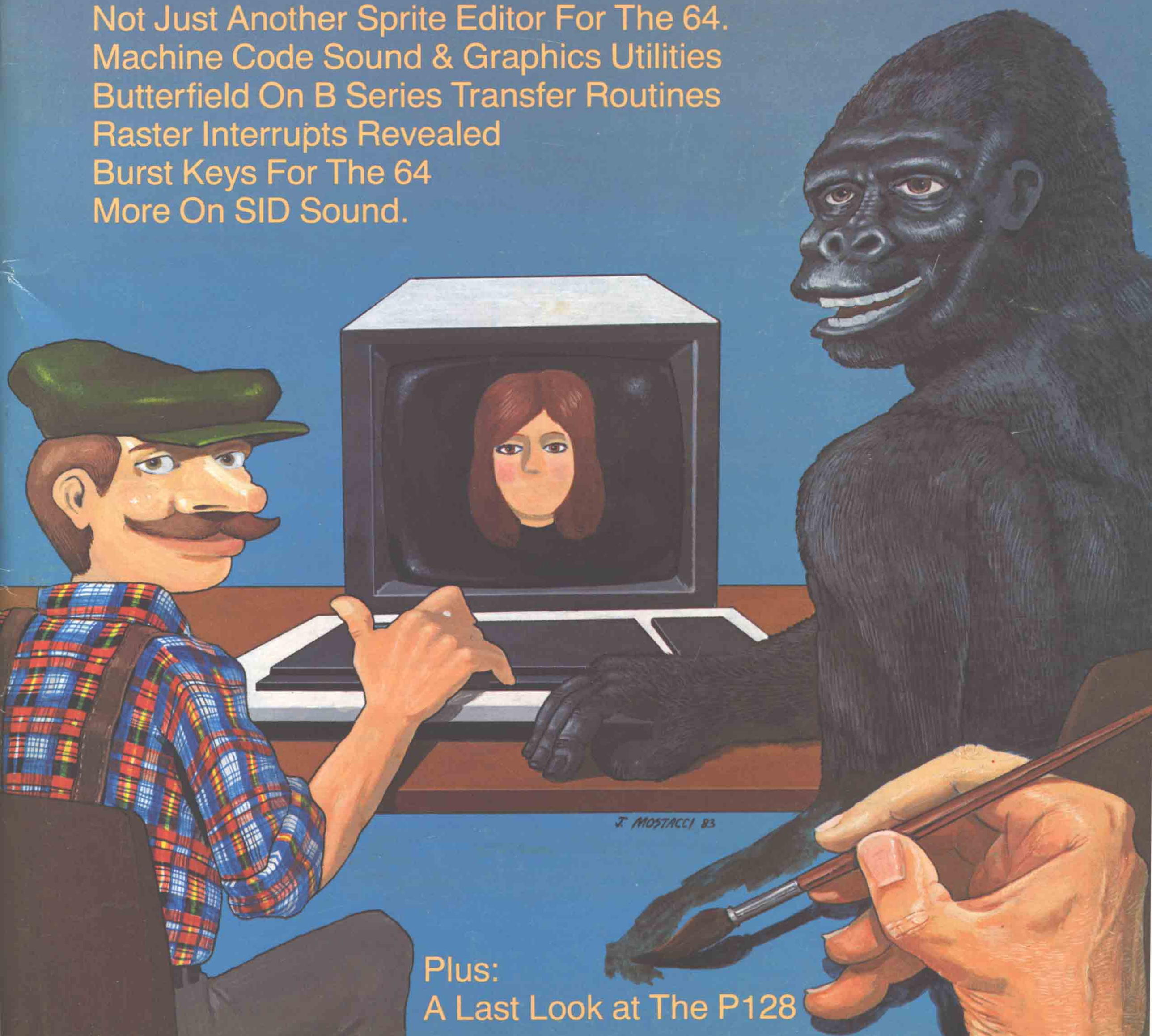
The Tech/News Journal For Commodore Computers Vol. 4.

July 1983

Issue 04
$2.95

Second Class Mail Permit Pending

Not Just Another Sprite Editor For The 64.
Machine Code Sound & Graphics Utilities
Butterfield On B Series Transfer Routines
Raster Interrupts Revealed
Burst Keys For The 64
More On SID Sound.



Plus:
A Last Look at The P128

Richvale Telecommunications

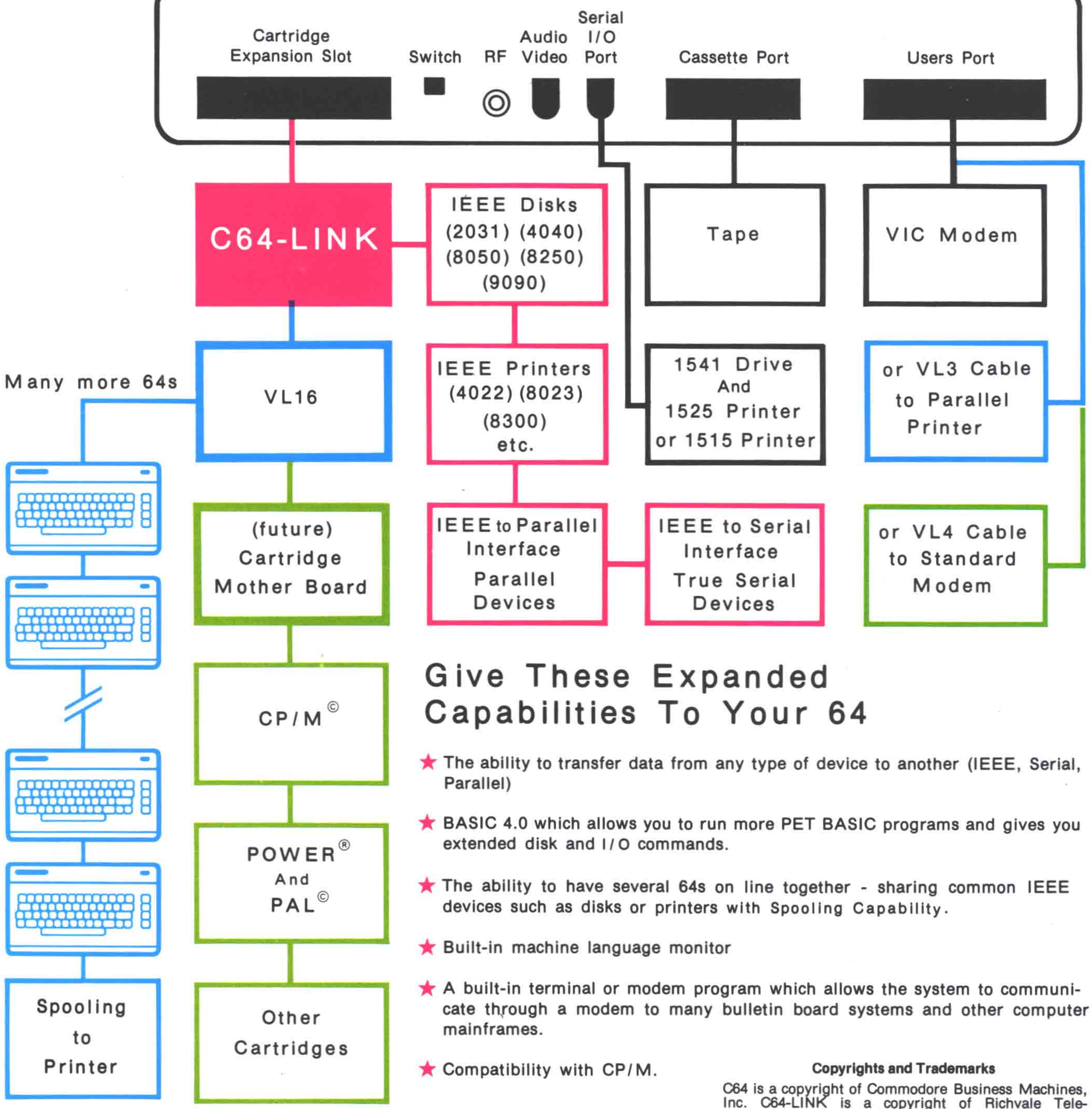
10610 BAYVIEW (Bayview Plaza)
RICHMOND HILL, ONTARIO, CANADA L4C 3N8
(416) 884-4165

$185 Canadian

C64-LINK©
The Smart 64

Also available for VIC 20

RTC RTC



Give These Expanded Capabilities To Your 64

- ★ The ability to transfer data from any type of device to another (IEEE, Serial, Parallel)
- ★ BASIC 4.0 which allows you to run more PET BASIC programs and gives you extended disk and I/O commands.
- ★ The ability to have several 64s on line together - sharing common IEEE devices such as disks or printers with Spooling Capability.
- ★ Built-in machine language monitor
- ★ A built-in terminal or modem program which allows the system to communicate through a modem to many bulletin board systems and other computer mainframes.
- ★ Compatibility with CP/M.

Contact your local Commodore dealer or RTC.

Copyrights and Trademarks

C64 is a copyright of Commodore Business Machines, Inc. C64-LINK is a copyright of Richvale Telecommunications. CP/M is a registered trademark of Digital Research. POWER is a trademark of Professional Software. PAL is a copyright of Brad Templeton.

The Transactor

Volume 4 Issue 04

Editorial 3
News BRK 4
Bits and Pieces 15
Transbloopors 24
CompuKinks 25
Computer Fair Report 26
The WordPro Book of Tricks 27
The MANAGER Column 30
Superkey–64 33
Re–Decoding The 8032 Keyboard ... 36
Making Friends With SID, Part 3 38
Sound Help 41
Sprite Palette For The 64 42
Commodore 64 Graphics Utility 47
Raster Interrupts On The 64, Part 1 .. 50
The P128: A First Look, A Last Look . 57
B500 Transfer Routines 60

Advertising Section 64
Advertising Index 76

The Transactor
The Tech/News Journal For Commodore Computers

Editor
Karl J. H. Hildon

Advertising Manager
Kelly George
416 826 1662

Contributing Writers
Greg Beaumont
Dave Berezowski
Jim Butterfield
Gord Campbell
Mike Donegan
Donna Green
Fred Hambrecht
Paul Higginbottom
Dave Hook
Eike Kaiser
Peter Lear
Bill MacLean
Darren Spruyt
John Stoveken
Brad Templeton
Don White

Production
Attic Typesetting Ltd.

Printing
Squire & Carter Ltd.

Artwork / Cover Art
John Mostacci

Program Listings In The Transactor

All programs listed in The Transactor will appear as they would on your screen in Upper/Lower case mode. To clarify two potential character mix-ups, zeroes will appear as '0' and the letter "o" will of course be in lower case. Secondly, the lower case L ('l') has a flat top as opposed to the number 1 which has an angled top.

Many programs will contain reverse video characters that represent cursor movements, colours, or function keys. These will also be shown exactly as they would appear on your screen, but they're listed here for reference.

Occasionally programs will contain lines that show consecutive spaces. Often the number of spaces you insert will not be critical to correct operation of the program. When it is, the required number of spaces will be shown. For example:

print" flush right" – would be shown as – print"[space10]flush right"

Cursor Characters For PET / CBM / VIC / 64

Down	– q	**Insert**	–
Up	–	**Delete**	– t
Right	–	**Clear Scrn**	–
Left	– [Lft]	**Home**	– s
RVS	– r	**STOP**	– c
RVS Off	–		

Colour Characters For VIC / 64

Black	–	**Orange**	– A
White	–	**Brown**	–
Red	–	**Lt. Red**	– V
Cyan	– [Cyn]	**Grey 1**	– W
Purple	– [Pur]	**Grey 2**	– X
Green	–	**Lt. Green**	– Y
Blue	–	**Lt. Blue**	– Z
Yellow	– [Yel]	**Grey 3**	– [Gr3]

Function Keys For VIC / 64

F1	–	**F5**	– G
F2	– I	**F6**	– K
F3	–	**F7**	– H
F4	– J	**F8**	– L

The Transactor is published quarterly by Transactor Publishing Limited. It is in no way connected with Commodore Business Machines Ltd. or Commodore Incorporated. Commodore and Commodore product names (PET, CBM, VIC, 64) are registered trademarks of Commodore Inc.

Volume 4 Subscriptions: Canada $15 Cdn
U.S.A. $15 US.
All other $18 US.
Second Class Mail
Permit Pending

Back issues are still available for Volumes 2 & 3. Best of Volume 2: $15 Cdn., U.S.A $17 US., all other $19 US. Volume 3: $15 Cdn., U.S.A $17 US., all other $19 US.

Volume 4 quantity orders: subtract 35% on orders of 10 or more.

Canadian Distributor:
Access Computer Services
125 Martin Ross Ave. Unit 10
Downsview, Ontario
M3J 2L9
(416) 736 4402

or:

CompuLit
PO Box 352
Port Coquitlam, B.C.
V5C 4K6
604 464 3396

U.S. Distributor
Prairie News
2404 West Hirsch
Chicago, IL
60622
(312) 384 5350

Send all subscriptions to: The Transactor, Subscriptions Department, 500 Steeles Avenue, Milton, Ontario, Canada, L9T 3P7, 416 876 4741. From Toronto call 826 1662

Want to advertise a product or service? Call or write for more information.

Editorial contributions are always welcome and will appear in the issue immediately following receipt. Remuneration is $20 per printed page. Preferred media is 2031, 4040, 8050, or 8250 diskettes with WordPro, WordCraft, Superscript, or SEQ text files. Program listings over 25 lines should be provided on disk or tape. Manuscripts should be typewritten, double spaced, with special characters or formats clearly marked. Photos of authors or equipment, and illustrations will be included with articles depending on quality. Diskettes, tapes and/or photos will be returned on request.

All material accepted becomes the property of The Transactor, until it is published. Once released, Authors may re-submit articles to other publications at their own discretion. Other magazines, of any nature, are invited to copy material from The Transactor, provided that credit is given to the Author (when applicable) AND The Transactor. Failure to comply will result in changes to this policy. Solicited material is accepted on an all rights basis only. Write to the subscriptions address above for a writers package.

The opinions expressed in contributed articles are not necessarily those of The Transactor. Although accuracy is a major objective, The Transactor cannot assume liability for errors in articles or programs.

From The Editor's Desk

The Transactor – Reflecting

A year has passed since The Transactor was resurrected from eternal dismallness. . . wait a minute – a whole year? Sure doesn't seem like it! But when I took a look back at the events that have catapulted me into the publishing business and changed the shape of The Transactor permanently, I realized that a year is only enough time to collect your gear – the true climb lies ahead!

When we began the new format, our direction was based on a number of philosophies that have generated a lot of mixed responses over the year. It started with our commitment to avoid turning into just another magazine. In some respects we've suffered from our "left wing" approach. It takes time to get used to something different. But we still believe that the road we chose was the best, even though it may not have been the shortest route. Let's get specific.

The first thing many notice is the paper used to print the magazine – flat finish as opposed to glossy. Although glare is virtually eliminated using flat finish paper, the weight is about 40% heavier. As the number of pages goes up, so does postage. However, we think the extra is worth it. Further, nearly every page is characterized by "white space", an ingredient which we feel makes text appear less arrogant and intimidating. Most mags won't consider this. It means they can't cram as much onto a page so they end up paying writers for more pages, and they need to use more paper. I've been told by others in the industry that The Transactor won't make it with this approach. I hope they're wrong.

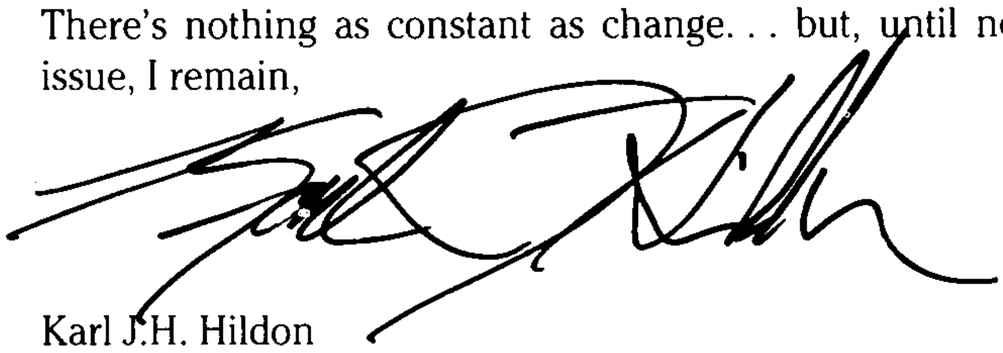
Starting from the front, the next trait several readers notice is the absence of a "Letters To The Editor" or "Questions and Answers" section. Well, they are there. . . just not as apparent. How many reader responses have you read that seemed somewhat less than a genuine answer? Many of the letters we've received have been answered, but indirectly. By taking a simple problem and elaborating means we can answer questions more thoroughly and eliminate any room for doubt and new questions on the same subject. Covering all of the possibilities is a philosophy we won't compromise.

Next is the Bits and Pieces section, my personal favourite. When I can see light at both ends of a paragraph, I just can't help but read it. So when I get my hands on a 4 line program, I can't resist publishing it. In fact, this section is probably the most popular part of the magazine. Some say they read even the parts that don't apply to them because it usually takes less than 5 minutes to assimilate each item. We believe that if we can help you get the most out of your computer in the least amount of time, your enthusiasm will not be threatened by the dreaded "this is gonna take forever" syndrome. Are we delivering?

Then there's the ad section, probably the most controversial philosophy of all. Why do we put every ad at the back? Several reasons. First of all it means that articles can be published without being interrupted by ads. Secondly it means ads can be printed with no interruptions from articles. Often I pick up a magazine just to look at the ads. Ads are important. The market information they contain is invaluable. So why does it get scattered all over the place? This way you can start from the back to read ads, or the front to read articles. The advertiser benefits too. Reading all the ads at once effects a more complete picture of the market, and an educated buyer is, in my opinion, the most valuable customer. Additionally, grouping ads means that at some later date a particular ad can be found quickly rather than paging through reams of meddley. All our readers seem to like the idea, and so far, our advertisers do too. In fact, we anticipate that advertisers will soon prefer being placed closer to the back than the front.

Over the next 12 months new changes are firmly in our plans: Our new Canadian and US distributors should help boost circulation; A full–time Art Director will be responsible for adding more colour and illustrations; Still more staff means our frequency will improve, hopefully to once a month in 1984. Aside from the magazine, Transactor Publications Ltd. is already negotiating several paperbacks. Long term plans include a multi–user bulletin board service, offered free to subscribers.

There's nothing as constant as change. . . but, until next issue, I remain,



Karl J.H. Hildon
Editor, The Transactor

News BRK

Volume 1 Bytes The Dust

Since 1978, The Transactor, Commodore's first in–house publication, has seen it's articles reprinted in four different languages in 7 countries and nearly a dozen other publications. A lot has transpired since "the beginning of time", but regrettably, the first volume of The Transactor is no longer available. Lack of demand and the respective age of the material just does not warrant any more reprints. Unfortunate too, because with it goes an era that saw the beginning of the microcomputer age as we know it – an age where "state–of–the–art" meant having a second cassette deck.

If you have ordered and paid for Volume 1, and have not received it, please don't panic. With the extra money, unless you object, we'll extend your subscription by 4 issues.

Volume 2 Transactors will be available while supplies last, but like Volume 1, it too is rapidly becoming prehistoric and it's doubtful it will see a printing press again. To alleviate that "Gosh, I've missed so much.." feeling that a lot you may be experiencing, many of the concepts covered in Volumes 1 & 2 will be reviewed in future issues of The Transactor, with updates for the new generation micros.

The Best of The Transactor Volume 3 is still quite current and will remain available until demand drops. Contained here are tutorial articles on the User Port (which has managed to stay consistent through the generations), the IEEE Port, and about 40 pages of VIC 20 material.

For the "complete set" minded, and in keeping with tradition, all issues of Volume 4 can still be obtained (see page 2 for details).

Gov't Promoting Software Industry

The Government of Ontario, through the Ministry of Industry and Trade, is anxious to promote the software industry in this province. Both the office of Procurement Policy and the Innovation & Product Development Branch of this Ministry are working in various ways to stimulate its growth.

The Office of Procurement Policy advises businesses in Canada, wherever possible, to buy Canadian, and keeps itself informed about manufacturers and suppliers of various goods and services, and potential customers.

Recently the Innovation & Product Development Branch has taken a special interest in the software industry in Ontario. It has begun to create a comprehensive resource list of all businesses active in software. The terms of reference are fairly broad. They are interested in all types of software, including programmed firmware. Micro, mini, and mainframe computers are areas of interest, as well as processors and controllers, such as those used in the automotive and microwave oven industries. The list will contain producers, importers, wholesalers, retailers, and lessors of software products.

The Ministry wants to hear from all sectors of the software industry, to learn what the status is, and how companies are faring. It is hoped that the survey will also bring out the potentials, so that informed recommendations can be made on what more can be done to promote growth. The survey team is particularly concerned that there are many small businesses which they will have a hard time reaching. They are asking all interested parties to contact them for further information. Call or write:

Mr. J.W. Vraets
Ministry of Industry and Trade
Technology Development Section
Innovation & Product Development Branch
900 Bay Street Hearst Block, Room 642 A
Toronto, Ontario
M7A 2E4 416 965 0157

Events

York Public Library Events

Looking for a course in microcomputers? York Public Library has become a veritable resource center for micro users in the Toronto area. In addition to their complete collection of books and magazines, York is continually presenting in-depth courses in Basic, machine language and other aspects. For more information contact:

York Public Library
1745 Eglinton Avenue West
Toronto, Ontario
M6E 2H4 416 781 5208

Toronto CES

Hunter Nichols Ltd. is pleased to announce the 8th annual Consumer Electronics Show, to be held at the International Centre in Toronto, July 24th to 26th, 1983.

Open to trade and related industries only, CES '83 will feature exhibits and displays by over 200 participating companies representing more than 500 brand names.

The industry is invited to see major brand names such as Sharp, Commodore, K-Tel, Mattel, Coleco, Atari, Toshiba, Alpine, Yamaha, Bose, Casio, and several others. For more info:

Gail S. Park
Consumer Electronics Sales Dept.
2282 Queen Street East
Toronto, Ontario
M4E 1G6 416 690 9666

Ontario Computer Day Camps

Microplace Inc. is a Canadian owned and operated company established by a group of parents whose background includes education, micro technology and communications. It was created specifically to prepare organizations, adults and children for the world of microcomputers.

The summer of '83 promises to be one of the most exciting ever. Micro technology is quickly becoming a part of everyday life. Those who will be most affected are the nations youth. Microplace is offering courses in levels from Beginner through Intermediate and Enriched in 24 locations across Ontario. The day courses are aimed at the 10–17 year age group with adult classes in the evenings.

The courses will use a Commodore 64 or Apple IIe computer according to preference. Each 14 hour course is $85 with additional hours available on weekends. For more:

Microplace Inc.
P.O. Box 162
Oshawa, Ontario
L1H 7L1
(toll free) 1 800 263 3727
(in Oshawa) 571 2837

4th Annual Pacific Coast Computer Fair

The Pacific Coast Computer Fair Association (P.C.C.F.A.), a non-profit organization, is presenting their 4th annual computer fair at the Robson Square Media Center in Vancouver, B.C. Show dates are Saturday and Sunday, September 17th and 18th between the hours of 10:00 am and 6:00 pm both days.

Events scheduled are User Group displays, Commercial Exhibits, Workshops, plus a full slate of speakers. The theme of this year's fair is "Computers – The Machine of the Future Today." Need more? Contact:

Penny Gross
P.C.C.F.A.
4100 St. Georges Avenue
North Vancouver, B.C.
V7N 1W8 604 585 1224

Software News

80 Columns For Your C64

RTC is about to release "Color 80" – software that turns your 40 Column Commodore 64 into an 80 Column. The utility redefines each character to half its normal width and uses the hi–res screen capabilities to display them. Full screen editing is not affected and no Basic memory space is consumed. The price? Just $34.95 in Canada, $29.95 in the US. For more call:

Richvale Telecommunications
10610 Bayview Avenue
Richmond Hill, Ontario
L4C 3N8 416 884 4165

6 Times Faster Cassette Operation

The *Arrow* from Skyles Electric Works is an inexpensive plug–in module for the VIC 20 or Commodore 64 that can increase the speed of cassette tape operations by 6 to 7 times. The package includes a machine language monitor and a complete set of text editing commands such as Search & Replace, Auto line numbering, multiple line Delete, and a Hex/Decimal converter. There is also an Append command and a high–speed tape positioning command that allows one to advance the tape by 16,000 byte intervals using the Fast–Forward button on the cassette deck. Price: $49.95 US.

A second version of *Arrow* comes with a built–in Assembler utility that works with disk or tape. Price is $89.95 US. For information call:

Skyles Electric Works
231E South Whitman Road
Mountain View, CA
94041 415 965 1735

Busiwriter And Busicalc

Busiwriter, also from Skyles Electric, is an inexpensive word processor for the Commodore 64. Features include 20 pages of text space, an enormous number of editing commands (with an interesting set of special characters to indicate text handling – Ed.) and is compatible with Commodore 1515 & 1525 printers, Epson, Qume, Diablo, NEC, C. Itoh (pronounced *See Eatow* for those as stumped as I was – Ed.), Starwriter, Prowriter, Okidata, Microline, Gemini–10, and others. Only $99.00 US.

Busicalc is an electronic spread sheet program that comes in four inexpensive versions. US prices are $49 for the VIC 20, $69 for the 64, $79 for 40 Column PET/CBMs and $89 for 80 Column machines. Call Skyles Electric Works (above) for more details.

WordCraft Receives ICP's Million Dollar Award

Dataview, a Colchester, England, based software publisher, and Peter Dowson, author of WordCraft, has just been presented with the software industry's most coveted award – The ICP Million Dollar Award for achieving 5 million dollars total sales of WordCraft, one of the earliest available wordprocessing programs for Commodore equipment.

The Million Dollar Awards were originated in 1972 by Larry Welke, president of International Computer Program Inc. The 1983 ceremonies were held in Scottsdale, Arizona, and at the Savoy in London, England.

Dataview celebrated the event by releasing a new and more powerful version of WordCraft. New features include a calculator facility, optional password security on disk files, additional modem support for modems other than the 8010 IEEE model, a parallel printer interface from the User Port, a "template" function to select fill file data based on specified criteria, and compatibility with alien files produced by other programs. More detailed manuals have also been added.

Europe:	N. America
Dataview	Cimarron
Portreeves House, East Bay,	2158 Hathaway Street
Colchester, Essex, CO1 2XB	Santa Ana, CA.
tel:(0206) 869414	92705 714 662 2801

DTL–Basic Compiler: New Versions

Dataview has also just released the DTL–Basic Compiler for the Commodore 64 and the new third generation Commodore machines. Contact Dataview or Cimarron.

Waterloo Basic For The VIC 20

Waterloo Computing Systems, in association with the University of Waterloo (Ontario, Canada), have made Structured Basic possible on the VIC 20. This extension to Commodore Basic offers the programmer IF THEN ELSE, DO WHILE, and REPEAT UNTIL capabilities as well as extra long variable names. . . up to 31 characters without ambiguity! Available now, no word on price. For details see your Commodore dealer or call:

Waterloo Computing Systems
158 University Avenue West
Waterloo, Ontario
N2L 3E9 519 885 1211

Business Pack For The VIC 20/C64

Superbyte Software has announced the release of their Business Pack for the Commodore 64 and VIC 20 (8K) computers. Combined within the package are the most widely used and popular business utility programs including:

- The Accountant – General Ledger, Income Statement and Balance Sheet
- A/R and A/P – Journal for current and paid accounts
- The Editor – Full feature word processor
- Spread Sheet Calc – Complete spread sheet/calculating program
- Business Inventory
- Checkbook Mate – Checkbook maintenance and writer
- The Mailman – Address file with sorting
- Plus 6 Utility Programs – Profit Margin Calculator, Business Calendar & Data Base, Program Evaluation Review Technique (PERT), Linear Regression Analysis, Depreciation, and Amortization programs

The package has full printer capabilities and comes complete with a detailed reference manual including program examples and a hard bound binder. Programming assistance in utilizing the software is also offered during specified hours. Introductory price for the entire package is $100.00 US. Call:

Superbyte Software
2 Chipley Run
West Berlin, NJ
08091 609 346 3063

Stock Helper For The C64

(M)agreeable software has released Stock Helper, a program written by a weekend investor for weekend investors to help keep tabs on the ups and downs of the stock market.

Stock Helper allows you to maintain a history of market prices, trends and indicators, display charts and graphs, and calculate moving averages over a 52 week period. Other features include input templates, choice of price form (decimals, fractions, or eights), sorting by name or market, and will accommodate stock splits and name or symbol changes, but refrains from giving you advice.

A version for the VIC 20 is planned for late summer. C64 version is available now for $31.25 US. or $38.50 Cdn including s&h. Call or write:

(M)agreeable Software Inc.
5925 Magnolia Lane
Plymouth, MN
55442 612 559 1108

Hardware News

Choosing A New Printer?

You owe it to yourself to see the largest selection of computer printers available under one roof at Guardian Data Products Inc. We offer a large and comprehensive stock of matrix, letter quality and line printers of all shapes, speeds, and prices. We also give you the technical assistance required to make them work with your computer. Lines include:

- Mannesman Tally
- Okidata
- F10 Printmaster
- Centronics
- Daisywriter
- Epson
- Gemini
- Prowriter
- Smith–Corona TP–1
- Diablo 630

All equipment is fully warranted in Canada and leasing terms are available.

Guardian Data Products
4699 Keele Street, Unit 15
Toronto, Ontario
M3J 2N8 416 665 4620

Auto Clock For VIC 20/C64

The Auto Clock is an intelligent real time controller for the VIC 20 or C64. The first thing the Auto Clock software does is set the internal clock of the VIC 20/C64 to the correct time. The computer can now turn itself on and off (thus, other devices as well) under program control.

In addition, there is 2K of CMOS RAM powered continuously by an on–board lithium battery. Using this, a "snap–shot" can be taken of any part of the VIC 20/C64's internal memory. Once done, the Auto Clock can power–down the computer and automatically restore that part of memory upon the next power–up. The snap–shot could even be a menu driver that would subsequently load another (larger) program from disk. Virtually an unlimited number of functions can be performed entirely without human intervention.

The unit plugs into the rear cartridge slot and comes with a 90 day parts and labour guarantee. Other features include an alarm, automatic time/date display anywhere on the screen, and a complete manual with memory maps! Price is $129.95 US. For more:

Progressive Peripherals & Software
6340 West Mississippi Avenue
Lakewood, CO
80226 303 778 1312

Starlighter Stage Lighting Controller

Also from Pregressive Peripherals comes the Starlighter; a highly sophisticated, computer assisted (VIC 20) phase control system for stage lighting equipment. With more features than physically possible on ordinary manual lighting controllers, it is still within reach of todays performing artists.

The control program is contained entirely in ROM, eliminating the need for tape or disk units to begin lighting control. The unit can control up to 32 channels and has up to 120 combinations or "scenes" on line simultaneously; 80 pre–programmed in ROM and room for 40 more that can be programmed by the user. These can be stored on tape or disk for later recall.

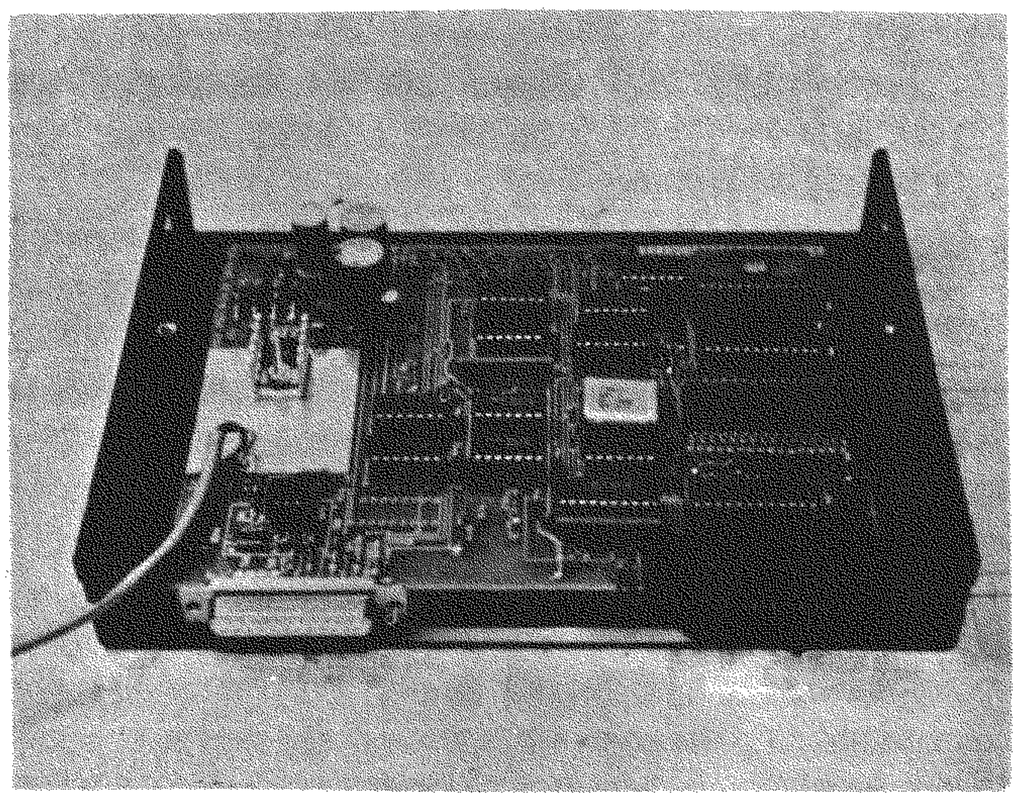
Other features include fading of individual channels or entire scenes, manual or automatic scene sequencing, a ten scene chaser program, and sofstart control which briefly pre–warms lamp filaments before allowing them to go to full brightness, thus greatly enhancing bulb life. A full colour video display provides graphic representation of all activity. The Starlighter is compatible with single phase, split–phase, and three phase AC line frequencies, and virtually all dimmer packs commercially available. Custom cables are available for special orders and custom PROMS can be programmed to suit unique applications. The Starlighter controller is $995 US plus cables and adaptors. For information, call or write Progressive Peripheral & Software above.

R6511Q Single Board Computer

This new single board computer uses the recently announced R6511Q microprocessor from Rockwell International. It uses an enhanced version of the standard 6502 instruction set and has an on–chip clock, asynchronous serial port, event counter, timer, 192 bytes of RAM and other features.

The basic configuration of the CmC SBC6511 includes the R6511Q CPU, power supply, address decoders, 1.8432 MHz crystal, serial port operation to 19200 baud, an 8 position CPU readable DIP switch, expansion ports and sockets for 2K of RAM and up to 8K of ROM.

Options include: an IEEE–488 (GPIB) interface; an RS–232 interface case; power pack; development board with monitor, trace, disassemble, etc.; assembler; digital and analog interfaces.



The basic board requires 9 to 12 volts AC or DC at 300 milliamps for power. Price is $189.00 US with substantial OEM discounts. Contact:

Joanne Akin
Connecticut microComputer
36 Del Mar Drive
Brookfield, CT
06804 203 775 4595

IEEE–488 Digital Output Module

Also from CmC is the BUSSter B64. The B64 is a 64 channel digital output module that's compatible with any computer that has an IEEE–488 bus (either built–in or added on). It accepts commands from the host computer through it's IEEE port, and activates 1 to 64 digital TTL lines that can be used to control non–standard peripherals, thus increasing the computers' interfacing capabilities. Easily programmed in BASIC. The standard version of the B64 is $495 US. including case and power supply. For more detail, contact Joanne Akin above.

C64 Motherboard Expander & E–P ROM Carriers

The RTC C64 Motherboard is ideal for C64 users with more cartidges than expansion slots. Of course, there's only 1 expansion slot but with this board up to 4 cartridges can be inserted. Four toggle switches allow easy selection of desired cartridge and reduce connector contact wear. Coming soon for the VIC 20. Price is $79.95 Cdn.

The E–P ROM Carrier for the Commodore 64 can accommodate ROMs, PROMs, or EPROMs in 2K, 4K or 8K sizes. Great for housing Basic extension utilities, game ROMs, and software protection ROMs. Price: $6.95 Cdn, with protective shell, $9.95.

Richvale Telecommunications
10610 Bayview Avenue
Richmond Hill, Ontario
L4C 3N8 416 884 4165

Modem Protection

Electronic Specialists' expanding product line now includes Modem protection. Kleen Line Security models are availabel for standard 4 pin telephone modular connectors (RJ–11) and the wider 8 pin connectors (RJ–45).

Intended to suppress damaging telephone line spikes caused by lightning, spherics, or phone switching gear, the Kleen Line system uses modern two–stage semi–conductor and gas discharge tube suppression techniques. An isolated ground is employed to isolate equipment from damaging lightning discharge current.

Model PDS–11 has suppression on red and green phone line (pins 3 & 4) with yellow and black brought straight through. Standard modular 4 pin telephone connectors provide simple, trouble free hook–up. Price: $56.95 US. (Call or write for free catalog. Request catalog #831)

Electronic Specialists
171 South Main Street, PO Box 389
Natick, MA
01760 617 655 1532

New Printers For VIC 20/C64

Both of the new printers from CardCo Inc. come complete with the well proven "card/print" interface installed internally, and should be available by late summer from selected computer dealers that carry the CardCo product line.

The "Cardprinter/LQ1" is a letter quality daisy wheel that prints at 14 CPS in either red or black. Features include bold, shadow, or underline print in normal or proportional modes of 10, 12, or 15 CPI. Suggested retail will be $599.95 US. and options will include a tractor feeder for $149.95, a cut sheet feeder with 200 sheet capacity for $199.95, and a keyboard for direct printing for $199.95.

The "Cardprinter/DM1" is a small dot matrix impact printer that will print 40 columns of text on standard 3 inch adding machine roll paper. The printer will feature a print speed of 50 CPS, full Commodore graphics, full high–res dot addressable graphics, and printing in either red or black through software control. Suggested retail will be $149.95 US. For further info:

CardCo Inc.
313 Mathewson Avenue
Wichita, Kansas
67214 316 267 6525

Parallel Interfaces For VIC 20/C64

Micro–Systems Development of Dallas Texas has announced two new products for its Interbus Series of VIC 20 and Commodore 64 interfaces.

MSD's *CPI*, a parallel interface for the C64 and VIC 20, solves several problems such as converting serial to parallel, providing ASCII conversion, automatic linefeeds, and program legibility. The CPI interface lifts the limitations imposed by the serial bus and will allow the user the option of using printers acquired previously.

Two program listing modes are provided. Since many printers do not support the codes that VIC 20s and C64s produce, the CPI will generate "tags" in their place. For

example, [YEL] is output for "change to yellow". For graphic characters the decimal value is printed.

The CPI connects directly to the serial port of the VIC 20 or Commodore 64 and is completely compatible for both requiring no special software.

MSD's *VPI* is a parallel interface for the VIC 20 only. This connects via the cartridge slot, is lower priced and has fewer features than the CPI. For more information on either product:

Micro Systems Development, Inc.
11105 Shady Trail, Suite 104
Dallas, TX
75229 214 241 3743

BusCard For The Commodore 64

BusCard, from Batteries Included, is a parallel bus interface that plugs into the cartridge slot of your C64. It initializes itself on power-up and uses no system RAM, ROM or I/O memory space. 3 ports: one for IEEE-488 devices, another for parallel printers, and a third for extending the cartridge slot.

Device number allocation is through an 8 pin DIP (Dual Inline Package) switch to eliminate software incompatible commands when transporting programs. Device #4 can be switched to IEEE, parallel, serial, or routed through an automatic PET ASCII to true ASCII converter before reaching the serial port. Devices 5 to 10 can be either Commodore serial or IEEE and 11 to 15 are hard wired to the IEEE port.

Also built-in are DOS "wedge" commands for disk communication, a machine language monitor and room for additional ROMs. Price is unconfirmed but under $200 Cdn.

Batteries Included
186 Queen Street West
Toronto, Ontario
M5V 1Z1 416 596 1405

RB5X: The Intelligent Robot

Micro Marketing Canada is pleased to announce the availability of the RB5X Robot in Canada. The RB5X, manufactured by the RB Robot Corporation in Golden Colorado, made its entrance into the home robotics market in the Fall of '82 and has been hailed as the world's first mass-produced, fully programmable, intelligent robot.

The RB5X has a sonar sensor for detecting objects in its path as well as 8 tactile sensors to signal retreat from possible collision. Another sensing device allows the RB to recognize low battery power and to seek out its charging unit. Upon reaching full charge it resumes activity. Other features include a horn, 8K of RAM expandable to 24K with 2K of EPROM, and card expansion slots for special electronic accessories.

The RB5X is programmable in BASIC which makes this robot especially useful for educational applications. Programs to control the robot can be written on virtually any home computer and then transferred to the RB's memory via a standard RS232 interface that's adjustable to 110, 300, 1200, or 4800 baud. Several applications have already been developed including navigational, home security and fire detection programs.

Among the options scheduled for future release are an off-the-shelf voice synthesizer module, voice recognition to allow for responses from verbal commands, a vacuum cleaner attachment, and a robotic arm able to lift, carry, and release light objects under direct program guidance or (depending upon the expertise of the owner) under self-learning guidance. Also coming are a trailer to haul small loads such as laundry or groceries and data telemetry circuitry to allow communication from one RB to another. As an example, an RB that has mastered its environment can transfer that acquired information to a second RB, eliminating the usual trial and error period.

The RB Corporation also publishes a monthly newsletter sent free to RB5X owners or on a subscription basis at $15.00 per year (prepaid). For more information contact:

RB Robot Corporation
14618 W. 6th Avenue, Suite 201
Golden, CO
80401 303 279 5525

or

Micro Marketing Canada
169 Inglewood Drive
Toronto, Ontario
M4T 1H8 416 484 9111

Also in the RB's future accessories are a projectile launcher for scaring the neighbors pets from your front lawn, and a rocket launcher and landing pad for the RB to retrieve first-hand weather information and report speed traps. A high-velocity paddle arm option will automatically activate at "bedtime" or during supervision of homework or piano practice. The particle beam laser attachment will come in handy for eliminating trash, illegally parked cars, unwanted

intruders, salesmen, relatives, . . . uhh please excuse this tomfoolery, I just couldn't resist – Ed. Incidentally, could a robot be lurking in Commodores' future?

The MicronEye Camera

The MicronEye Camera, from Micron Technology, is a camera lens combined with digital electronics to provide a video input interface for your Commodore 64. The lens is mounted on a tri–pod and has focus and F stop adjustments just like a normal camera. But with only a little bit of software, the MicronEye can transmit a picture directly to your high–res screen.

It comes in three configurations; an RS232 version that does not include software, and 2 other versions that do. All three include hardware, optics, and manual, and the MicronEye Bullet package is ready to use via the 64 cartridge slot. Prices start around $295 US. For more info:

Micron Technology Inc.
2805 East Columbia Road
Boise, Idaho
83706 208 383 4000

Commodore News

Salutations, Ed Kellow

Ed Kellow, President of Commodore Canada, is retiring at the end of June. Mr. Kellow started with CBM back when their main gig was typewriter service and repair. Does this mean no more golf tournies at Kleinberg?

Looks like James Dionne has been inflicted with the task of finding a successor. Shouldn't be too hard to find someone though, especially since all the tough ground has been broken. Stay in touch E.K., who's motto is, "not rain nor snow nor tall of grass shall keep me from my appointed rounds."

Bon Voyage Paul

Paul Higginbottom, Commodore Canada's "protoplasmic data base" for the past two years, and Commodore U.K.'s before that, is on the move again. This time to sunny Dallas Texas where he'll join Commodore's research and development team. Paul's presence has been a dominant force here in the Great White North, but his absence will surely be felt even more. Farewell, but Goodbye?. . . no way!

Financial Report

New York, New York – Irving Gould, Chairman of the Board of Commodore International Limited (NYSE:CBU), announced today (April 26/83) that Commodore's sales, net income and earnings per share for the company's third fiscal quarter ended March 31/83 are expected to be sharply ahead of results recorded in the comparable year–ago quarter.

According to Gould, "Commodore's sales, net income, and earnings per share in the recent quarter which ended March 31 are expected to be the best three month period in the company's history and more than 100% ahead of year–earlier results." In the third quarter of fiscal 82, Commodore reported sales of 82.1 million, net income of $11 M, and earnings per share of $.71.

Gould further noted that "Commodore's outlook for the current quarter and into fiscal 84 is excellent. The demand for Commodore 64s has far exceeded our most optimistic internal projections. As a result, production rates have been increased from less than 5000 units monthly in September 82, to over 20,000 units in December, and over 40,000 units in March 83. Similarly, sales of disks and printers are also excellent with demand exceeding production. As a result, we are again raising production levels of peripheral devices to meet that demand."
"In the aggregate," said Gould, "Commodore is now producing well in excess of 100,000 microcomputers per month, something that has afforded the company tremendous economies of scale."

Gould concluded that, "Commodore's future outlook has never been better and we expect further significant growth in the fourth quarter of fiscal 83 ending June and into fiscal 84."

The preceding was brought to you from our "so what else is new department". By the way, did anyone buy $1000 worth of Commodore stock back in '78. No? Good. Because you would hate to find out that it's now worth about $40,000. Of course hindsight is always 20/20 vision. – Ed

New Products And Prices at Chicago CES

This year's Chicago CES brought the announcement of a few new products and several price reductions on existing equipment. First the new stuff.

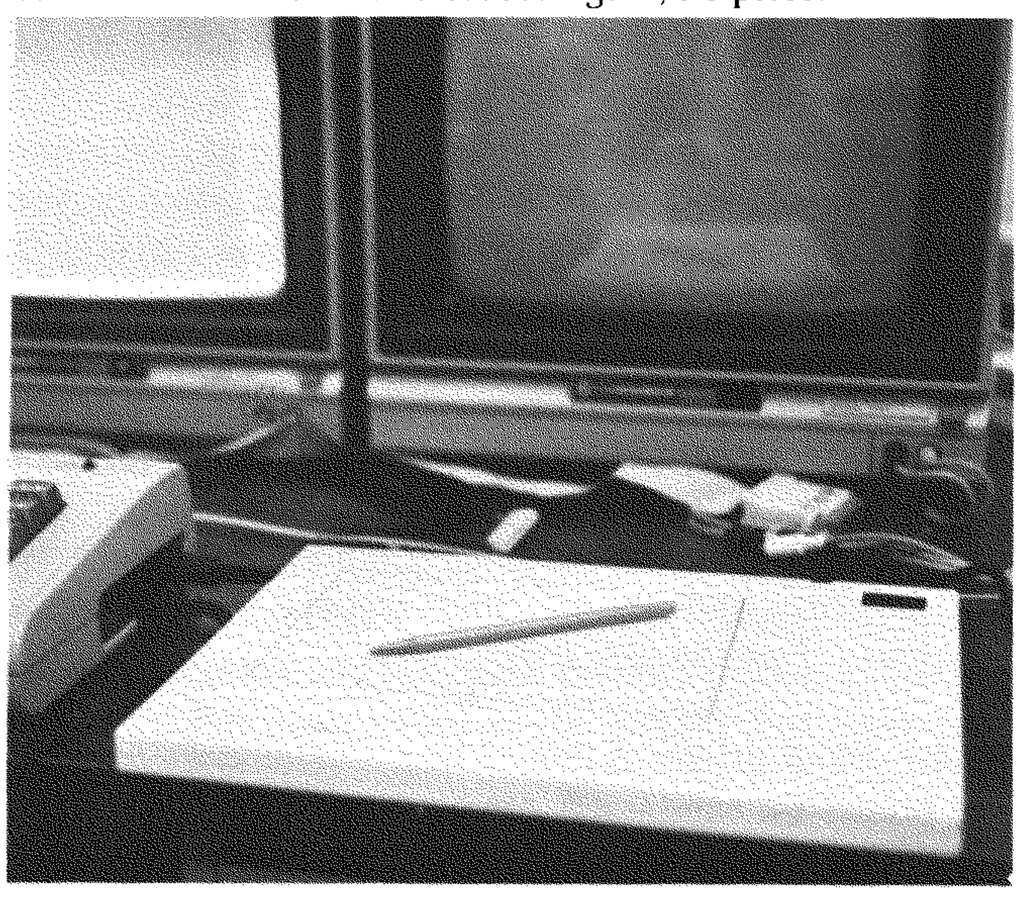
A new colour printer is due to be a Commodore product. The printer on exhibit did not display a Commodore part

number so it isn't necessarily the release version. For the demo it was reproducing a C64 screen done in multi colour mode. Seven colours on tap, but like most colour printers it was rather slow. No word on price.

Other new accessories included a "Mouse" type gadget like the one that comes with the Apple "Lisa". A Mouse is basically an upside-down trak-ball with a fire button. The trak-ball is better for games, but the mouse can be very handy in business software for controlling certain activities.

The Trak-ball was also released for public perusal. A Trak-ball is a must for some games like Centipod. Briefly, it's just like those things you use to clean your hands at the bowling alley, except it has two fire buttons. Unclear what the second one is for, but it also has an ON/OFF switch when you don't want it to interfere with the keyboard. No price available.

A Graphics Tablet was on display too. It plugs into the paddle port of your 20/64 and allows for direct pen to screen design. Nice solid unit. . . hopefully some solid software will come with it when it's released. Again, no price.



Overall the show was a success for the helmsmen at CBM. Last year they vowed to be the number one hardware company. This year they're after the number one software spot. Commodore is taking on piles of programs written by software houses across the country. Many are being sold in the form of modules (ie. A/P, A/R, GL, etc.) and most, including the C64 Manager, are going to be sold for $49.95 US. Also, look for game cartridges to dip as low as $9.95.

Based on the trend towards lower prices, and competition of course, Commodore has lowered the prices of most of their products from previous generations. The 8032 started at $2495 Cdn. About a year ago it was reduced to $2195. Now it's been dropped to $1395! The 4040 is now $1495 Cdn, no big deal, but the 8050 was cut to $1695 (from 2495 Cdn). The 8250 took a blow to the price bucket too – $1895 in Canada.

The SuperPET has always been the subject of rumours and controversy. Will it last? Will it be discontinued? Looks like not. Commodore introduced the SPET at $2795 Cdn. At CES they announce the new price of $1895. Some CBM brass believe the SuperPET is Commodore's biggest sleeper and expect sales to take a rise this fiscal year in the educational market. Thinking is based on improved product awareness – as more schools check out 20s and 64s, it's believed they'll want to know more about the other lines and consider additional purchases.

PET Emulator Now Public Domain

Commodore has released version 2 of the PET Emulator to all dealers for distribution to their Commodore 64 customers at cost or free of charge. The program modifies the C64 such that it operates identically (in most respects) to the BASIC 2.0 PET, thus opening up the vast collection of PET programs to 64 users. See your dealer for further details.

Critical List Update

It seems that some of the products mentioned here in previous issues will not reach the market. Those being The Max machine and the 'C' Series formerly known as the 'P' Series formerly known as the PET II.

The Max was to be the entry to the extreme low end of the market designed to compete with video game machines. But with the price of the VIC 20 dipping to as low as $48 in some states, the Max lost its purpose in life and was shelved.

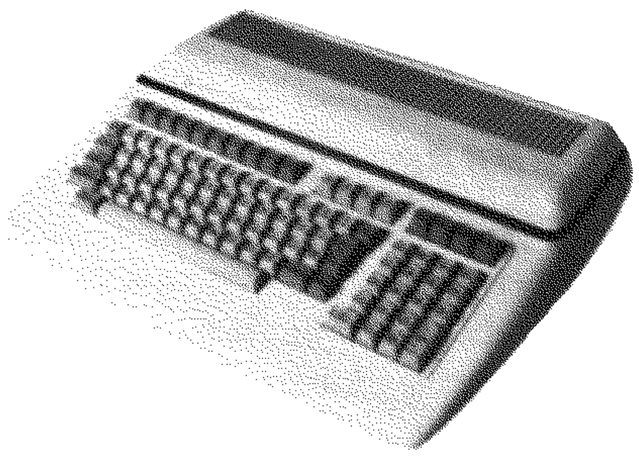
The 'C' Series machine was a member of the third generation family of machines. Main features were BASIC 4.0, IEEE

and RS232, 128K of RAM, numeric pad and 10 function keys, 40 columns with colour. In essence it was a C64 with 128K RAM using the 6509 processor as opposed to the 6510 in current 64s. Reasons for discontinuing the project are possibly due to an undefined market for the product. See the article on the P128 this issue for more details.

New Product Name Changes

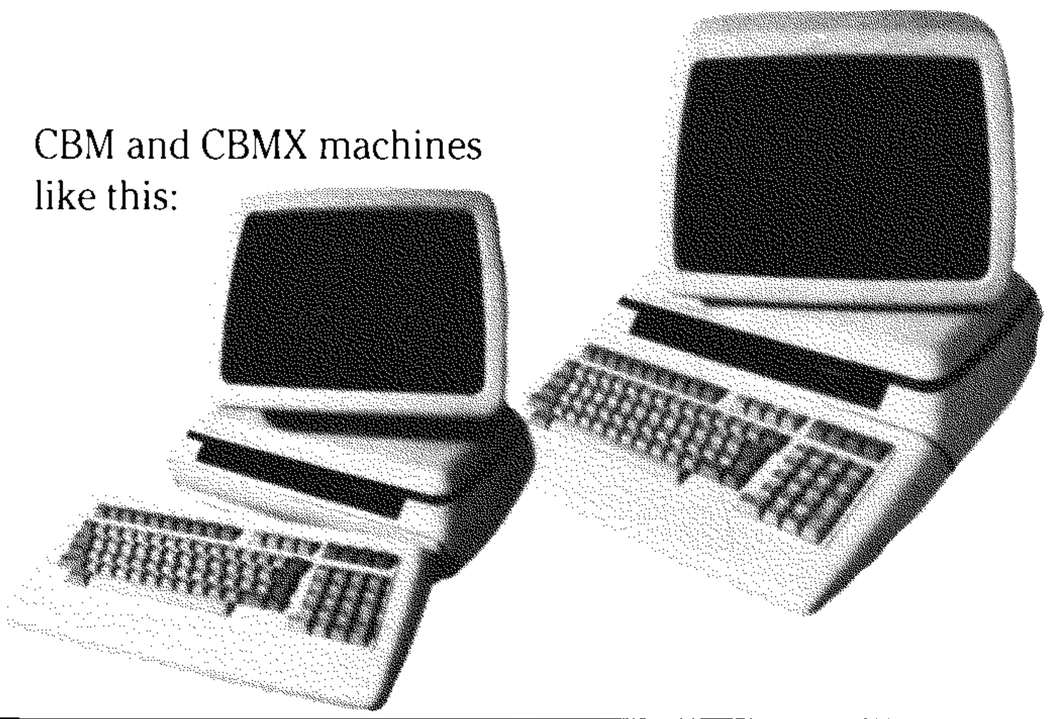
Here we go again folks. Nameplates for the new third generation or 'B' Series machines have undergone surgery once more. Basically there are 8 products in the queue, some of which have already been released in limited quantities. All have BASIC 4.0, 80 column output, both IEEE and RS232 interfaces, but no colour.

The latest product names are the B128, B256, BX128, BX256, CBM128, CBM256, CBMX128, and CBMX256. The 8 machines can be split into two groups; The first 4 are just the keyboard and main logic board. The other 4 have the integral swivel/tilt monitor with space for built-in disk drives. The numbers represent the amount of RAM in K bytes and the "X" stands for "extra processor". This will be Zilogs Z80 or their new Z8001 for running CP/M based software.



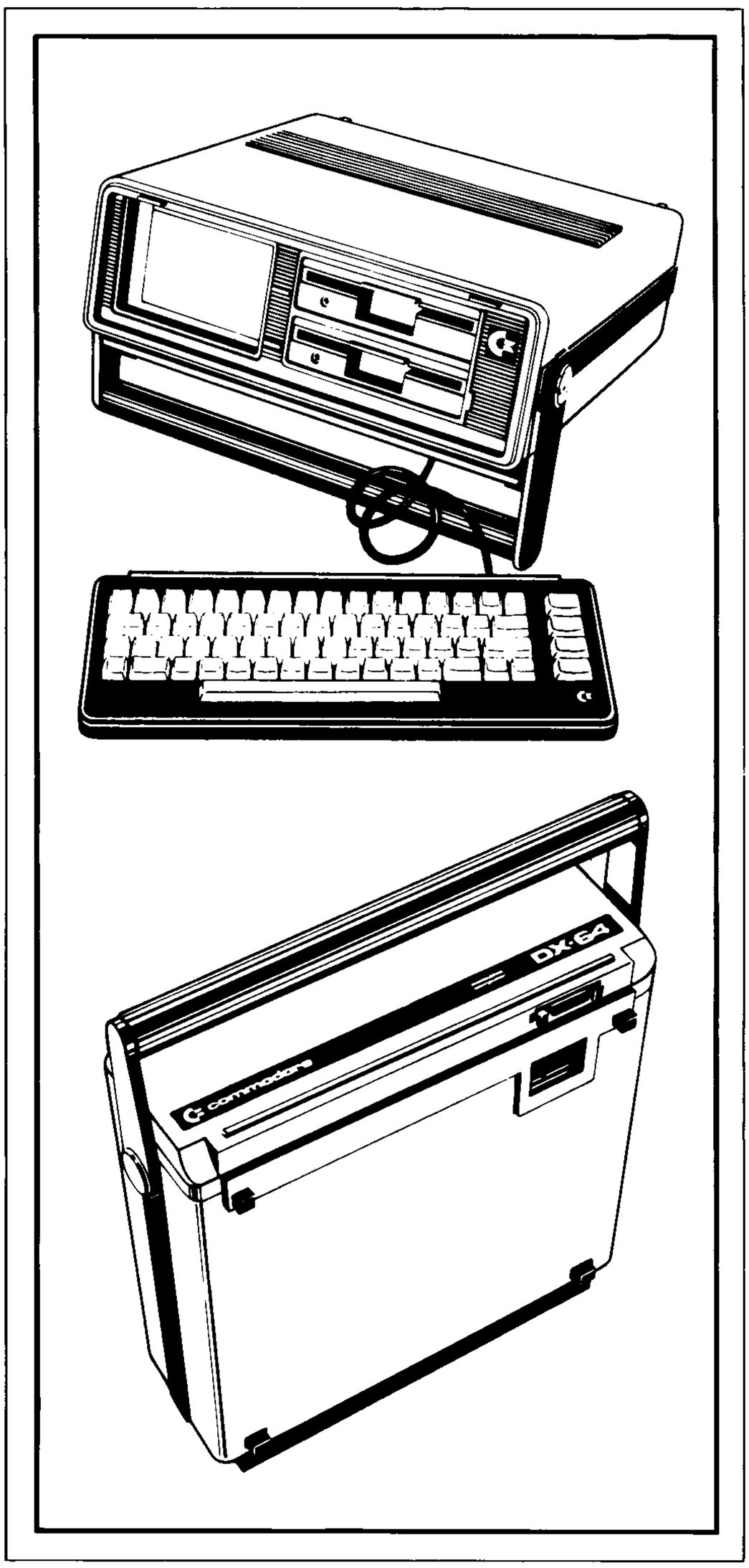
B and BX machines will look like this,

CBM and CBMX machines like this:



For the CBM Series, Commodore has designed a double sided 4040 as the integral disk drive. 8250 type drives were on the drawing board at last report.

The SX-100, which used to be the SX-64, is now the "Executive 64". Just to refresh. . . this is the one with the 5" colour monitor, disk drives, and keyboard all in one carry--all case - like the Osbourne. Reported prices are $995 US. with one drive, $1195 with two drives. The unit has no cassette port but all the rest are there. Cartridge port is on top, serial port is extended at the back for chaining periphs.



Free Calc Result Packages

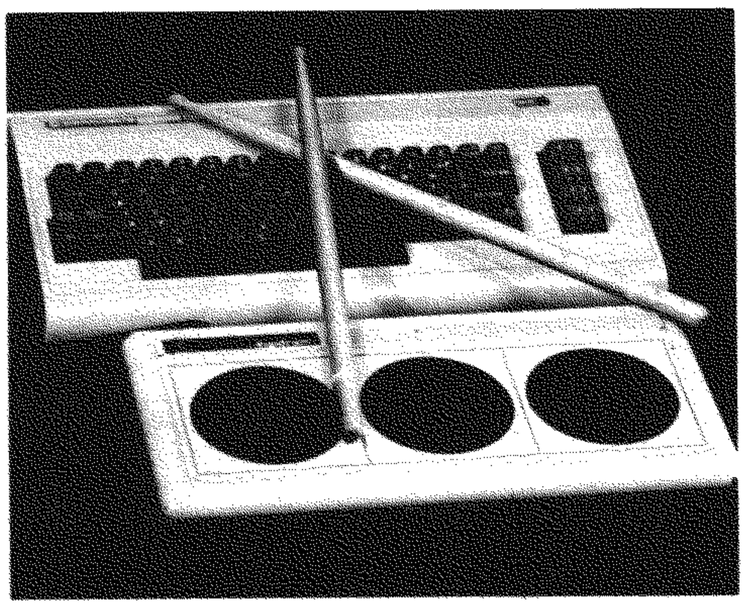
Commodore has announce that every purchaser of a Commodore 8096 will receive a Calc Result program free of charge.

The 8096 comes with 96K of RAM, a capacity which gives enough versatility for most business needs. With a combination of the 8096 and Calc Result comes a powerful planning and forecasting tool, a three dimensional spreadsheet that does "what-if" calculations and converts complex operations into easy-to-read graphics. Contact any authorized Commodore dealer for details.

Commodore Digidrum Adds New Dimension

Imagine an electronic drumset controlled by a computer, with volume control and sound generated through a TV set, monitor or stereo system. . . but priced as low as a video game cartridge! At a recent CES show, Commodore demonstrated a new low priced three-pad electronic drum set called DIGI-DRUM(tm) compatible with either the Commodore 64 or VIC 20.

This new peripheral will plug into the expansion port and comes complete with special software which lets the user simulate a snare drum, base drum, and "high-hat" cymbal with startling realism. [thanks to "SID" Higginbottom]



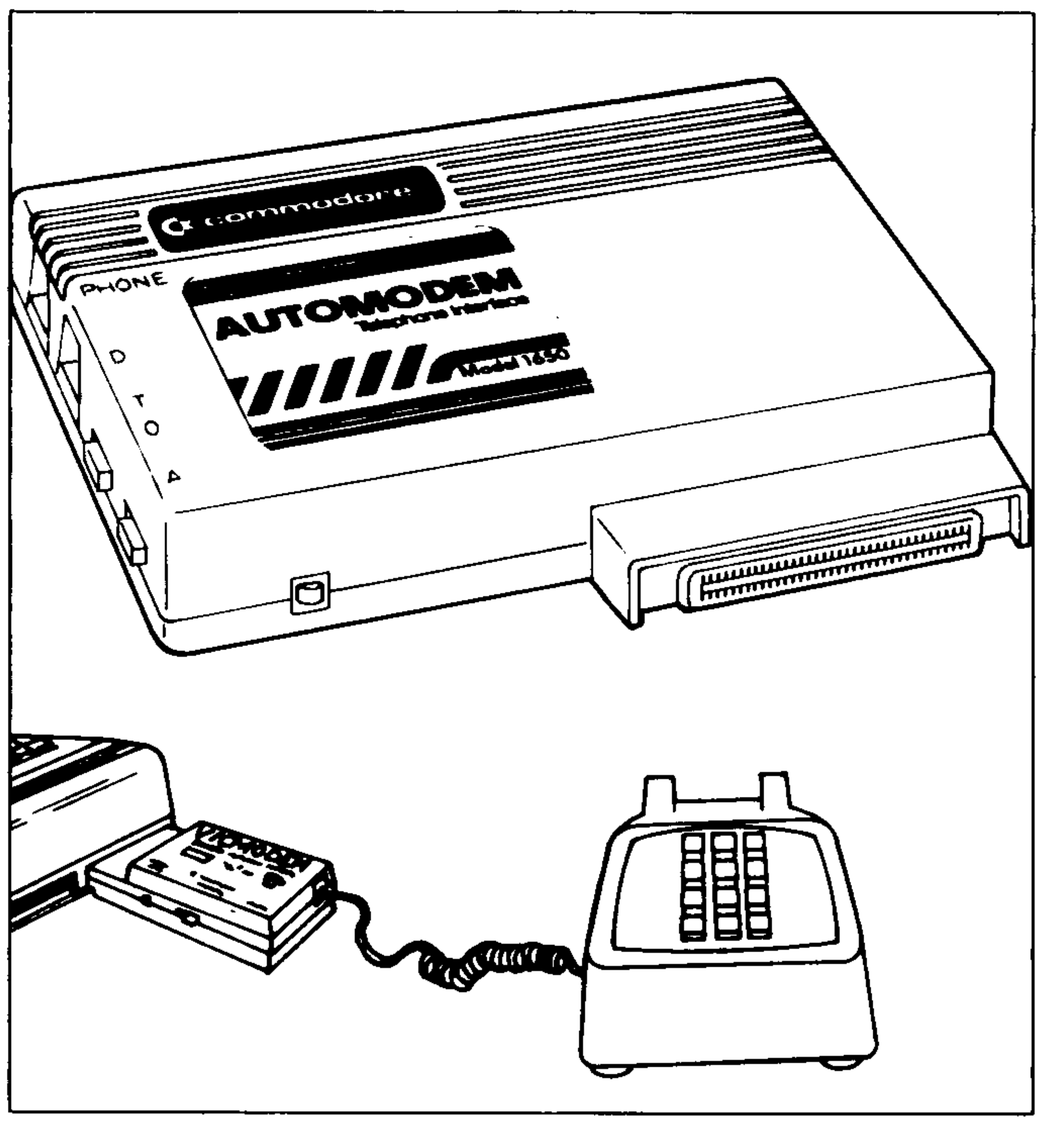
The combination of computer and Digi-Drum visually displays 3 animated drums on the screen which performs with each drum stroke. Other sound/screen simulations can be programmed by the user. Sequences can be created, replayed, and saved to disk or tape for later replay. No word on price, deliveries planned for Spring/Summer 1983.

New VICModem Package

Commodore is now shipping a rather conclusive package for VIC 20/C64 user interested in the vast sphere of communications. The package contains: the popular VICModem; a white standard Northern Telecom telephone with Commodore Logo; a cassette tape containing terminal software for the 20 or 64; one free hour of time on the CompuServe timesharing network; one free hour on the Dow Jones News/Retrieval Service; a three month membership in Comp-U-Store; and savings on subscriptions to other information services including Marketscan and Info Globe. Contact any Commodore dealer for details.

C-1650 Automatic Modem

As well as the VICModem, Commodore will offer the Auto-Modem for use with VIC 20 and Commodore 64. Like the VICModem, it will connect via the User Port, and will have Auto Dial, Auto Answer, and selectable number of data bits and start/stop bits. Comes with a free hour on CompuServe. No word on price or availability.



Bits and Pieces

One Line Squiggle

Squiggle was one of the very first programs written for the PET 2001 back in 1977. It didn't really do very much except draw a continuous pattern of lines on the screen. Since then a lot has been learned about Commodore Basic and Squiggle has been rendered many refinements. Even though Squiggle still doesn't do very much, it now does it in only one line of Basic.

```
100 print " Sq  1 line squiggle "
110 c$=chr$(34)
120 print " qqq type <return> 3 times
130 print " q new
140 a=548 : b=peek(57345) : if b=75 then a=167
150 if b=86 or b=220 then a=204
160 print " qqq 1pO " right$(str$(a),3) " ,0:x=4*
    rN(1)+1:fOi=1to10*rN(1):?ml(" ;
170 print c$ " qQ [Lft] " c$ " ,x,1) " c$ " Q [Lft] "
    c$ " ;:pO " right$(str$(a+1),3) " ,1 " ;
180 print " :nE:gO1 "
    190 print " run
    200 print " sqqqqq " ;
```

As you can see there is much more than just one line here. However, this program prints another program after some decisions are made about what type of computer is underneath it (Lines 140 & 150). It will work on any Commodore machine, although BASIC 2.0 machine users will want to ensure that A = 167 as we couldn't find a 2.0 machine to test it on. Also, it's best to enter the program using Upper/Lower case, but run it in Graphics mode. . . the program doesn't switch itself.

The program won't fit into one edit line so abbreviations are used. The mnemonic [Lft] is of course a Cursor Left character. You might LIST the program after it's entered to see just how much can actually be squeezed into one line using abbreviations.

VIC 20 and Commodore 64 users could make several additions I'm sure. A random colour changer would be an interesting mod no doubt. Once again, we'll be most pleased to see any new versions but I can't imagine them fitting onto one line anymore. . . or could they?

Late N.B.: Can anyone explain why after an extended run on a Commodore 64 the RUN/STOP key is disabled? RUN/STOP–RESTORE is also locked out. Most peculiar.

Invisible Colours

No, this is not a new feature. . . just a reminder that by using the Commodore Logo key instead of the CTRL key in combination with the colour keys across the top, you can obtain the other 8 colours even though they aren't shown on the key fronts. Of course this only applies to C64 users since the 20 only has 8 colours. The corresponding colours are:

CTRL 1 = Black	(0)	Logo 1 = Orange	(8)
CTRL 2 = White	(1)	Logo 2 = Brown	(9)
CTRL 3 = Red	(2)	Logo 3 = Lt Red	(10)
CTRL 4 = Cyan	(3)	Logo 4 = Gray 1	(11)
CTRL 5 = Purple	(4)	Logo 5 = Gray 2	(12)
CTRL 6 = Green	(5)	Logo 6 = Lt Green	(13)
CTRL 7 = Blue	(6)	Logo 7 = Lt Blue	(14)
CTRL 8 = Yellow	(7)	Logo 8 = Gray 3	(15)

Can someone think up a "jingle" using the first letters of these colours? Some of your may know this one: Bad Boys Rape Our Young Girls But Violet Gives Willingly. It's a great way to remember your resistor colour codes. A similar one for these would become a world standard!

Miscompulations

The following miscellaneous compilations for the 64 come from Howard Strasberg (age 12) of Toronto, Ontario.

People always want their BASIC program to be unable to list, and unable to break in to. Well, if you start the first line with a REM followed by a shifted "L", when you type LIST, the computer will display part of line one and then a ?SYNTAX ERROR! However, like everything, you can still list it by taking out line 1 or whatever your first line may be. This is a way to prevent listing before the program is run.

To prevent listing after the program is run, you simply disable LIST in your program. This is done by:

```
poke 774, 0 : poke 775, 141
```

This will make it so that when you type LIST, you get a clear screen and a READY. Also, when STOP and RESTORE are entered, these two POKEs remain so your program still cannot be listed. In order to list your program type:

```
poke 774, 26 : poke 775, 167
```

This next one will disable STOP/RESTORE and make your listing go crazy. Load a program and type:

```
poke 808, peek(808)-5
```

Now try a LIST. Obviously to return to normal:

```
poke 808, peek(808)+5
```

Another good thing to do is disable SAVE so people cannot copy your program once it is run. This is done by POKE 818,0. However, you must also disable STOP/RESTORE as this will enable SAVE. By the way, with the disable LIST Pokes, the program can be SAVEd but still not listed.

Ever got your finger worn out from banging on the same key over and over again to draw a long line or something? Well, now your problem is solved. All you have to do is POKE 650,128 and now you can hold down the keys! No more worn out fingers either!

Cathode Ray Tubing

PRINTing is a term that has grown with computers since even before the dawn of their concept. For without the need for some form of output data, what use would we have for them? But unlike the pioneers of computed data who relied mainly on bulky motor driven output, most of us have come to depend on the CRT screen for most data display. A printer enters the picture only when it becomes necessary for output to be transported, stored over some period of time, or consolidated in large quantities. Indeed these limitations of the CRT may one day be history, but at the present it is all too apparent that the CRT is only the next step in the evolution of the printer. The concept of PRINTing has only been carried over to a new media and the full potential of the CRT has become lost in the limitations we have unnaturally imposed on ourselves from the mechanics of printers.

The next couple of routines will attempt to dissolve some conditioning that us humans have allowed. Although we can't put CRTs in our pockets (yet!), we need not treat them like printers with no moving parts. And besides, who says that printing goes left to right anyways!

```
100 a$ = " now is the time for the crt to come of age "
110 cols=80 : c=len(a$)
120 if c and 1 then a$=a$+chr$(32) : c=c+1
130 for j=1 to c/2
140 print tab(co/2-j)mid$(a$,c/2+1-j,1)chr$(145)
150 print tab((co/2-1)+j)mid$(a$,c/2+j,1)chr$(145)
160 next : print : print
170 run
```

The above centers each line output. Adjust COLS to the size of your screen. The program here merely demonstrates a technique. In actual practice, A$ would be read from disk or DATA statements and line 170 would be a RETURN from a GOSUB to this routine. Line 120 ensures that A$ contains an even number of characters. CHR$(145) is a Cursor Up and could be replaced with the quotes-mode character.

Have you tried this yet? Good. If you didn't like it, you might be impressed by reversing it. If you did, try this anyways.

```
130 for j=c/2 to 1 step -1
```

Your printer would hate you for trying this one

```
100 a$=" now is the time for the crt to come of age "
110 c=len(a$)
120 if c and 1 then  c=c+1
130 for j=1 to c : x=c*rnd(1)+1
140 print tab(x) mid$(a$,x,1)chr$(145)
150 next : print tab(1) a$ : print : run
```

Applications? CRTubing really only comes in handy when you've got a batch of text destined for the screen and the PRINT statement just doesn't seem very intellectually stimulating. Instructions for a game fall into this category and there's no need to slow it down so the reader can keep up. Besides, it might get them to read the instructions the first time.

Combomands

This next batch of weirdisms are key combinations that invoke commands on the Commdore 64. Essentially they throw a curve at the hardware that lies just beneath the keyboard.

This one from Craig MacIntyre of Toronto. Press SHIFT, "?", and the space bar simultaneously. It's much like hitting some erroneous keys and a Shifted RUN/STOP, except it's a lot tidier in the finger department.

These are from Darren Spruyt of Gravenhurst, Ontario. Darren wouldn't reveal the outcome to us in his letter, so we're passing on the cessation. Here's part:

". . .press the plus, minus, and the pound, and hold down. Now do it again, this time holding down the shift or Commodore key. Neat! Now, holding the left shift, press and hold the keys two, three and four. One final item: hold down the left shift and press and hold down the Q, W and E keys. Have Fun!"

Number Numbing

This has to be the fastest, cheapest wordprocessor of all time. It has full screen editing, and can handle files easily over 400 lines. It's compatible with all CBM printers, and most others, and it's fast! The program uses no memory:

```
Basic 4/2 - open 4, 4 : cmd 4 : poke 19, 32 : list
VIC20/C64 open 4, 4 : cmd 4 : poke 22, 35 : list
```

That POKE there in the middle makes the machine omit line numbers when LISTing. It works on the screen too. All parameters for LIST work as normal, you just won't see the line numbers.

This creates some interesting possibilities. Using one of any Basic Aid type programs, you instantly have a text editor with enough flexibility to do most jobs. To enter a line, simply type a line number followed by text. Need more lines? Use your Renumber command. And most Basic Aids offer Delete, Search, and Search with Replace. To see it on the screen, just give the POKE without OPENing, etc. Adding a line or a ?SYNTAX ERROR turns it off.

TIming The Commodore 64

While on the subject of timing, several Commodore 64 users in the U.S. and Canada have noticed a slight discrepancy in the accuracy of TI and TI$; the internal clock. So *slight* that it can add up to about 2.4 seconds per minute!

The problem stems from the factory. Commodore makes two versions of the 64 that have only one difference - the crystal oscillator that generates the system clock frequency. The crystals are different to accommodate the different types of colour TV sets. Namely, PAL sets such as those made in the U.K. and Europe, and NTSC (North American Television Transmission Standards Commission) sets like those found in North America. (C64 users elsewhere should check with a service department to see which they have)

Unfortunately, televisions are rather picky things. Without getting too much into detail, NTSC sets operate around a frequency of about 1.02273 MHz. In PAL sets the number is about 0.98525 Mhz. Although Commodore installs the proper crystal into 64s depending on destination, they don't make two different sets of ROMs to adjust the TI timer accordingly on power up which, naturally, is also affected by the clock crystal.

The crystal cannot be replaced without snafuing your video output but there are two solutions here. . . the one you pick

will depend largely on how much time and money you have to waste. First, you could buy a voltage converter and a round trip ticket to anywhere in Europe for you and your 64. Now run a TI$ dependant program against your timepiece and you'll notice the problem has gone away.

The second comes from Greg Beaumont of Toronto. Try this:

poke 56325, 66

The clock is updated during interrupts. Interrupts are controlled by Countdown Timer A in CIA 1 of the 64. The low order byte of the timer (56324) continuously counts down from $FF to 0. Each time the low order byte reaches 0, the high order byte is decremented by 1. The high order byte of the timer is also a latch. The number you POKE into the latch is the number that the high order byte begins counting down from. When the high order and low order both reach 0, an interrupt is generated on the IRQ line.

Normally, the value 64 is placed in the latch at power up. The interval at which each interrupt will occur can be calculated by:

Frequency / (256 * latch) / 60 = X

Where X = the interval in 60ths of a second. When the latch value is 64, X works out to:

1.02273 e6 / (256 * 64) / 60 = 1.0404 60ths

Omit the divide by 60, and the number of clock updates per minute comes to 62.42. This is too many. Therefore, the latch value must be increased so that Timer A must go through more countdowns in order to generate an interrupt. Using a latch value of 66 results in an interrupt interval of about 1.008 60ths of a second. This is about as close as you can get short of adjusting the clock frequency.

For those that could care less about the clock, this leaves open some other possibilities. For example, is your cursor too slow, or maybe you'd like LIST to be a little less hasty. Try POKing 56325 with values lower and higher than 66. This will cause more or less (respectively) interrupts to occur and affect the general speed of your 64 based on the number of times the interrupt routines are serviced.

DATAdjuster

Can you say Restoreshjiblinkuhwitz? Then you should have no trouble putting these next Pokerismettes to good use. As you may have have guessed, it involves the somewhat unpopular RESTORE command.

As you all know, RESTORE adjusts the DATA pointer so that the next READ command will pick up the first element of the first DATA statement in your program. However, sometimes it would be nice if the DATA pointer could be directed to the line of your choosing. A RESTORE with an optional line number parameter would have been perfect for this. Although it wouldn't be hard to write a machine code utility to do this, these next POKEs will accomplish the same result.

Consider the Current DATA Address at decimal locations 65 & 66 in your VIC 20/C64 memory maps (62 & 63 in BASIC 4.0/2.0). This pointer will always be somewhere between the beginning and end of your BASIC text. It actually starts at the beginning of BASIC and, upon a READ command, goes forward through text until it finds a DATA statement. If it gets to the end of text without finding one, an ?Out of DATA error results.

There is another pointer that will always be somewhere within BASIC text and that is the CHRGET pointer within the CHRGET subroutine. This pointer lies at decimal 122 and 123 (119 & 120 in BASIC 4.0/2.0). This is convenient. Now all we need to do is get the CHRGET pointer as close as possible to the DATA we wish to READ next, and then transfer its contents into the Current DATA Address pointer. Like this:

```
 10 rem restore x simulator or datadjuster
 20 print "which block of data ?"
 30 input "1, 2, 3, or 4"; x
 40 on x gosub 100, 200, 300, 400
 50 read a$ : print a$
 60 if a$ <> "end" then 50
 70 print : goto 30
100 poke 65, peek(122) : poke 66, peek (123) : return
101 rem 4.0/2.0 users: remember to change 65 to 62,
        66 to 63, 122 to 119, 123 to 120
110 data "blk", "wht", "red", "cyn", "end"

200 poke 65, peek(122) : poke 66, peek (123) : return
210 data "one", "two", "three", "end"

300 poke 65, peek(122) : poke 66, peek (123) : return
310 data "yuk", "blech", "brap", "end"

400 poke 65, peek(122) : poke 66, peek (123) : return
410 data "buzz", "whir", "click", "crunch", "end"
```

Notice that every block of DATA must be preceded by a duplicate of the adjuster subroutine. When the POKEs are

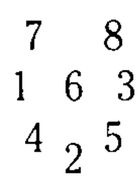
finished, the Current DATA Address pointer will be somewhere just prior to the RETURN command. But that's ok. When READ is executed, BASIC goes searching for the next DATA statement.

Another variation might be to save the value of the pointer into two variables, say DL and DH, which might be pointing into the "middle" of a DATA statement. Then you could go READ some other block of data and return the pointer to the address stored in DL and DH to continue READing from the place you left off.

New 64 Video Port

As mentioned in an earlier issue, Commodore is now shipping C64s with a new 8 pin video connector. Reasoning? It seems some users have been plugging their video cables into the power connector, and we can't have that.

Although there are three extra pins, only one of them has been connected to anything internally. The others don't change. The new port configuration is:



Face View

1. Luminance
2. Ground
3. Audio Out
4. Composite Video
5. Audio In
6. Chroma
7. No Connection
8. No Connection

As you can see, only Chroma has been added to the connector outputs. This will only be of use to those with monitors that have Chroma input connectors. The pattern of the connector doesn't change much either. The only incompatibility you need be concerned with is inserting 8 pin plugs into the old 5 pin ports.The only question I have is, won't the power plug now fit into the video port? If it does, and you do, watch it. Your video chip will have one more feature; air-conditioning. Colour coded stickers & tape might be a wise safety precaution under potential circumstances. – Ed.

New VIC 20 Power Supply

VIC 20s are now being shipped with the same power supply as the C64. New VICs will also get the new power plug which means that they too are succeptable to the problem described above. But with a little common sense, it won't happen.

The old transformer continued to dissipate power even with the VIC turned off. This one actually shuts off with the VIC (like the 64), thus prolonging transformer life.

Three Blind Noughts

With machine language becoming so popular, many "hybrid programs" are being written. A hybrid is a program combining Basic and machine language. Often the machine code begins right where the Basic text ends. But where IS the threshold point? Usually we won't need to know this, until us curious types want to look at the code that follows Basic.

Since there are no pointers set up by Basic to indicate this spot, the only way to find it is to look for the 3 consecutive zeroes that mark the end of Basic text. This is the point where the LIST function terminates. This type of work invariably requires a machine language monitor to scan memory, so if you don't have it built–in, you should arrange to load one (eg. the appropriate version of Supermon) before you load the program you wish to examine. But it's generally pretty hard to estimate the size of text in bytes from looking at the size of the LISTing, so where do you start?

The first method we all seem to try is a memory dump starting from the address where Basic text begins. As the hex listing scrolls by, we franticly scan each line hoping to catch a glimpse of three zeroes. But this can be rather tough, since they may not be all on one line. By the time empty memory rolls around, your eyes are declaring war against your brain, and of course you missed them three cursed zeroes. Feel like trying again?

Next, we could write a program to OPEN the program file on disk, read the start address, and increment it each time a byte is read until we count three zeroes in a row. Then after you calculate the address into hexadecimal, you LOAD the program, SYS to the monitor, etc, etc. But this doesn't work for tape, and besides, how many of you are actually going to have this utility handy when you need it? It would probably be faster to re–write instead. BLECH!

How 'bout using ROM routines to find it. All Commodore machines have a ROM routine that rebuilds the Basic text chain links every time a line is inserted or removed. As the routine executes, it maintains an address in RAM workspace that is continually updated until the routine reaches the end of text. Once done, this address can be viewed with the monitor.

Enter Murphy's 467th law. "A byte that contains important information will be destroyed before being allowed to examine it." The only problem with this routine is that it leaves this address in a place that the operating system uses a lot. So by the time "READY." is printed, the address is clobbered and we're no further ahead. Therefore we need a second SYS to transfer the address to a safer spot. Here are both together, and they must be entered on the same line separated by a colon: (Note – the ":SYS4" or ":SYS8" at the end is to get you into the monitor for the .M command next, but this could be entered separately.)

```
Basic 4.0– sys46262 : sys62792 : sys4
Basic 2.0– sys50242 : sys62729 : sys4
VIC 20     sys50483 : sys54762 : sys8
C64        sys42291 : sys46570 : sys8
```

Enter Murphy's 468th law. "A byte you wish to transfer to a safer place will be clobbered before it gets there." This is exactly what the second SYS does. But only to the low order byte of the address. Fortunately the high order stays in tact. (Murphy doesn't have a law for addresses, just single bytes) So the second SYS (BASIC 4/2) transfers only the high order part, although it could be "backed up" to get both. The VIC20/C64 SYS's unavoidably transfer both, but again the low order is invalid.

Now, using the monitor, you can display the page number that contains the end of text marker.

```
BASIC 4/2   .m 00db 00db [or peek(219)]
VIC20/C64   .m 004f 004f [or peek(79)]
```

Although you'll still have to do some eyeballing to find the 3 blind noughts, at least you can single it down to one page of memory. If anyone has a better approach, short of writing a program, or finds a single SYS that gets the whole address, it would make a splendid update.

Retina Wrencher

If you are chronically sane and wish to stay that way, then don't, I repeat, DON'T enter this program. Of course if you know someone you'd like to see go right around the bend, just run this program for them. But don't dare look at it! Or you too will suffer the wrath of the diabolical Retina Wrencher!

The program is the handywork of Richard Evers, Toronto Ontario. It works only with machines that use the 6551 CRT Controller chip. Commonly, these are the 8032, 8096, SuperPET, and the Fat 4032.

```
10 input " S enter a number between 0 and 13 " ;a
20 print " use the Shift key to terminate program "
30 for z=1 to 1500 : next : print " S "
40 poke 59520, a : z=1
50 for c=32768 to 34767 step 7
60 poke c, int(rnd(1)*255) : next
70 for d=0 to 255 step z
80 poke 59521, d
90 if peek(152) then print " n " : run
100 next
110 for e=255 to 0 step -z
120 poke 59521, e
130 if peek(152) then print " n " : run
140 z=z+1 : goto 70
```

Caution: The Zero option may cause damage to your CRT if left running for too long. Try it, but hit your Shift key right after you've had a chance to see it. To bring you this spectacular display, the zero option strains the CRT yoke somewhat. Leaving it run for too long would be like driving you car on the highway in second gear; eventually something breaks. When Shift is pressed, the screen will stay blank for a moment so don't panic. Afterwards, the characters will appear a bit smaller but will resume normal size shortly. Although The Transactor can assume no liability for damage, both Richard and I will be extremely disappointed to hear of any, especially as the result of malicious intentions. In the past, programs like this have not been released for this reason, and if there is but one report, there will be no more.

The other options are all quite harmless (to the machine that is). However, some don't do very much. If, after the initial pattern is displayed, there is no apparent activity, terminate it with Shift and try another. But once again, avoid zero, or you'll drive yourself and your machine insane.

Supermon Notes

Un–beknown to many is the Supermon "P" command. The command exists in the BASIC 4.0 version as well as the VIC 20 and 64 renditions. It's used to send continuous disassembly of code to a printer. The printer must first be activated with:

```
open 4, 4 : cmd 4
```

Then give a SYS 8 (SYS 54386 on BASIC 4.0 Supermon) to get back into the monitor. Now follow "P" with Start and End Addresses and hit Return. All code between these addresses will be sent to the printer in disassembled format. Actually, any output activity that normally occurs on the screen will now be diverted to the printer. This includes "R" for Register Display, "M" for Memory Dump, and "D" for Disassemble except this will only send X number of disassembled lines where X equals the number of lines normally output on your screen. To deactivate the printer, issue a:

```
print#4 : close 4
```

As mentioned in an earlier issue, POKE 53281, 12 will change the background colour to grey offering better contrast, especially to those with C64s connected to B&W screens. But this means you first gotta exit Supermon, give the POKE, and enter the monitor again. Why not do it from Supermon? Greg Beaumonts' favourite first off is:

```
:d020 fb fc <return>
```

You might also try . . .F3 FF. . . for a fairly clear contrast or play with your own combinations.

You might be asking, "what colours are FB, FC, F3 and FF?". You could effect the same colour changes by using 0B, 0C, 03 and 0F. The leading 'F's are there mainly to avoid a '?' mark prompt when you hit return. The ? is generated when Supermon reads the location you just changed and finds it doesn't match your selection. This is because the colour registers are only 4 bits wide – the lower four bits. Remember, there are only 16 colours. . . why have 8 bits when only 4 are needed. The upper 4 are effectively non–existent and an open connection in hardware, as a rule, is always logic hi. The Fs fool Supermon into a correct verification.

Machine Code Delay

Usually when we decide the only alternative is machine language it's because BASIC is just too slow. But sometimes machine code can be too fast. The following bit of code is a common subroutine for inserting delays.

```
        ldx  #$00
delay2  ldy  #$00
delay1  iny
        bne  delay1
        inx
        bne  delay2
        rts
```

As you can see there are 2 simple loops, one (using the Y register) nested within the other (using .X), and will yield about 0.25 seconds of delay on the average PET/VIC/64. But what if 1/4 second still isn't enough. You could find a countdown timer in some I/O chip to use, but this can make programs very machine dependent. JSRing to it over and over will do, but you'll need another register to increment and we've already destroyed two of them which means you'll need to use a free memory location. How 'bout a third outer loop and we'll free up the X and Y registers while we're at it.

```
        php          ;save Carry flag (optional)
delay   clc          ;clear Carry
        pha          ;# times thru inner 2 loops
        lda  #$00    ;initiate secondary loop
delay2  pha          ;save secondary loop
                     count
        lda  #$00    ;init primary loop
delay1  adc  #$01    ;primary loop
        bcc  delay1  ;loops 255 times
        clc          ;un-set Carry
        pla          ;recall sec loop count
        adc  #$01    ;increment it
        bcc  delay2  ;255 complete?
*       pla          ;recall # iterations (C=1)
        sbc  #$01    ;decrement
        bcs  delay   ;Carry still set?
        plp          ;restore Carry (optional)
        rts
```

With the extra code, the two inner loops will add up to a little more than .25 seconds, excluding time spent on any interrupts that may occur in the process. For fine tuning, try adjusting the second LDA operand with numbers slightly above zero. For now though, we'll assume the delay is still .25s.

Before calling the routine, the Accumulator (.A) is given a value that will be the number of iterations, plus 1, of the two inner loops. Plus 1 because the two inner loops will always execute at least once. Therefore:

Effective Delay = (.A + 1)*.25s
Minimum Delay = (0 + 1)*.25s
Maximum Delay = (255 + 1)*.25s = 64s.

The two inner loops escape to the point marked * when the Carry flag is set. Thus there is no need to do an SEC before the SBC. When the PLA at point * receives a zero, subtracting 1 results in a clear C flag and the delay ends. Otherwise, the new "outer loop" iterations value is stacked at DELAY, and the two inner loops are repeated.

Saving and restoring the P register (processor status) is optional. Since the routine is totally dependent of the Carry flag, you may wish to preserve it in case it contains "hot data" prior to your delay.

Sure, it's a little longer, but 12 bytes is a small price to pay for a more versatile solution. Plus, it will save a lot of headscratching when it's unclear whether .X and/or .Y need be preserved.

Flag Stacking

Machine language programmers are no doubt well aware of the value of the instruction "PHP"; PusH Processor status. The Status register (often called the "P" register) is, for all practical purposes, a byte stored within the microprocessor itself. This byte, like any other byte, has 8 bits. However, each bit is used to represent the occurence of some condition present in the computer as the result of a previous operation. For example, if the operation of subtracting one byte from another yields a value of zero, the bit that represents the Zero flag (or "Z" flag) is set to "1". Subsequent code might then test this flag in a decision making process for transferring execution.

Since there is only one P register, it often becomes necessary to store it for future reference while some other code is executed that may affect and change P register flags. In the previous item on delays, the routine starts with a PHP and ends with a PLP to restore the state of the Carry flag in case the code following the delay is dependent of the C flag. Remember, dependent can mean two things; dependent of Carry Clear, OR, dependent of Carry Set – a condition that cannot necessarily be assumed upon exiting the subroutine.

The same holds true for all the other flags in the P register. One that requires a particular amount of attention is the "I" flag or Interrupt Disable flag. When I is set (ie I = 1), an IRQ (Interrupt ReQuest) will be ignored by the microprocessor. Hopefully I won't be set for too long as this would disable any keyboard servicing and your machine goes into never neverland.

Let's consider the following sequence of code. It has absolutely no meaning except to demonstrate an effect:

```
. . .   ;code leading up to..
PHP     ;save P reg on stack
SEI     ;disable IRQs
. . .   ;some bunch of code. . .
PLP     ;recall P reg
```

After the "PLP", will the IRQ be accepted or ignored by the CPU? Those of you that said, "I dunno", are absolutely right! Without knowing the condition of the I flag prior to the "PHP" instruction, there is no way of giving a correct Yes or No answer. But this is good. Because no matter what it *was* it will be returned to that state by PLP, naturally! By using PHPs and complimentary PLPs, one can set or clear flags for a certain stretch of code, subroutine, etc., and return the flags to their previous state to make decisions based on previous conditions.

N.B. I wrote this piece after examining a routine that appeared to be missing a CLI instruction (CLear Interupt disable). When I took the bag off my head, I saw why it wasn't necessary. What does seem necessary now is another article on interrupts. Look forward to one! – Ed.

Arithmetickling

This next item has absolutely nothing to do with your computer, but it will get you thinking. Stretch out and see if you can spot the glitch. If your skin starts crawling away, you can use your machine to iron out the wrinkles, but try it without first.

Step 1	$a = 2$
Step 2	$b = 1$
Therefore	$a = 2b$
Multiply by	$(a - b)$
Then:	$a^2 - ab = 2ab - 2b^2$
Subtract	ab
Then:	$a^2 - 2ab = ab - b^2$
Multiply by	$(a - 2b)$
Then:	$a = b$
So:	$2 = 1$
Subtract 1	$1 = 0$

But this is absurd! Uh huh, it is. I'll be leaving now.

SuperPET Bits

In the previous issue we posed the question, "How many bits in a SuperPET?". Did anyone try it? How many came up with the answer 1,181,104? What, you say there's more?

APL Character Set

Want to access the APL character set of your SPET? Richard Evers of Toronto has this:

```
poke 59520, 12 : poke 59521, 48
```

To get back, use "Escape–Reverse–n".

ACIA Status Handling

The 6551 ACIA (Asynchronous Communications Interface Adapter) is an extremely powerful and efficient chip when communication between the SPET and other devices is required. But it has come to our attention that there is a bug in the 6551. This bug comes in the form of a discrepancy that separates the 6551 from all other 6500 series ICs.

In all other MOS interface chips such as the 6520 PIA, 6522 VIA and the new 6526 CIA, when data is received into an input register, a flag is set in another register usually known as a status register. Often this flag is "tied" to the IRQ line so that an interrupt is generated. The microprocessor then goes examining status registers to see which chip caused the interrupt, and then services it accordingly. PET, VIC, and C64 cassette tape routines work this way. When the data register is read by the CPU, the status register is cleared automatically by the internal hardware of the chip. The IC essentially prepares itself to receive and indicate the arrival of new data.

In the 6551, unlike the others, the status register is cleared not when the data register is read, but when the status register itself is read. This can be potentially hazardous during simultaneous input and output of data on your communication lines.

In programs built to handle these situations, output is usually done in the main stream of your program, for example when spooling from disk (when sending from the keyboard, characters usually only go one direction at a time and this problem will probably not occur). Input is usually handled by an interrupt routine invoked by the IRQ line. But IRQ can be generated by a number of sources. It is up to the interrupt routine to determine which source the interrupt request is coming from. The following will demonstrate how the interrupt routine could get snafued if the IRQ comes from the 6551.

```
inoutreg = $eff0
status   = $eff1
cmdreg   = $eff2
ctrlreg  = $eff3
```

Notice that the Input reg is the same as the Output reg.

In our example we'll assume that the Command and Control Registers are properly configured and that a disk file has been opened for spooling to the communications line. Typical code might look like:

```
testout lda status    ;get status reg
        and#16        ;last char sent?
        beqtestout
        jsr  $ffcf    ;get char from disk
        sta inoutreg ;put char in acia to send
        jmptestout
```

The branching tight loop at "TESTOUT" is testing the Transmitter Data Register Empty flag, bit 4 of the Status register. When bit 4 goes high, the character has been sent and the loop is exited. Now another character can then be queued for output.

But let's say a character comes in from the line during this spooled output. In typical asynchronous communications with, for example, a host mainframe, the character could be an X-Off (transfer off) character instructing your SPET to stop sending. The 6502 always completes the current instruction before servicing an interrupt. If the X-Off were to come in during the AND or the BEQ, no problem. Your interrupt routine could examine the status reg where it would find the 6551 was the source of the interrupt. However, if the character arrives during the LDA instruction, the interrupt would still occur, but your interrupt routine would find that the flag in the 6551 status register has been cleared by LDA STATUS. This means you cannot naturally determine that an interrupt has been generated by the 6551. Since 40% of the time spent in this loop is on the LDA instruction, this could potentially occur 4 times out of 10!

There are two solutions here – one software, the other hardware. In software, your interrupt routine would examine the 6551 status register first. If this were not the source of the interrupt you would continue by testing the other chips. But if all are tested and there is still no chip claiming responsibility, then you must assume it was the 6551. This will work until you start communicating at speeds of 19,200 baud. Testing all other potential sources for generating an IRQ and then servicing the 6551 by default may not be fast enough.

The better solution is in hardware. Instead of having the 6551 generate an IRQ when characters are received, it would generate an NMI interrupt. NMI is not used for anything else in the SuperPET, so why not. Simply disconnect the 6551 from the IRQ trace and hook it up to the NMI trace. Now simply point the NMI vector at your code to handle incoming data. Since NMI doesn't do anything else, you need not even test for the source of the interrupt – just got directly to the 6551 service routine. With this modification, the bug in the 6551 will give you no further trouble.

Transbloopors

This section, although hopefully not regular, will cover publication errors and/or improvements for articles in previous Transactors. If you find a significant mistake in our mag, please let us know so we can pass it on. Ed.

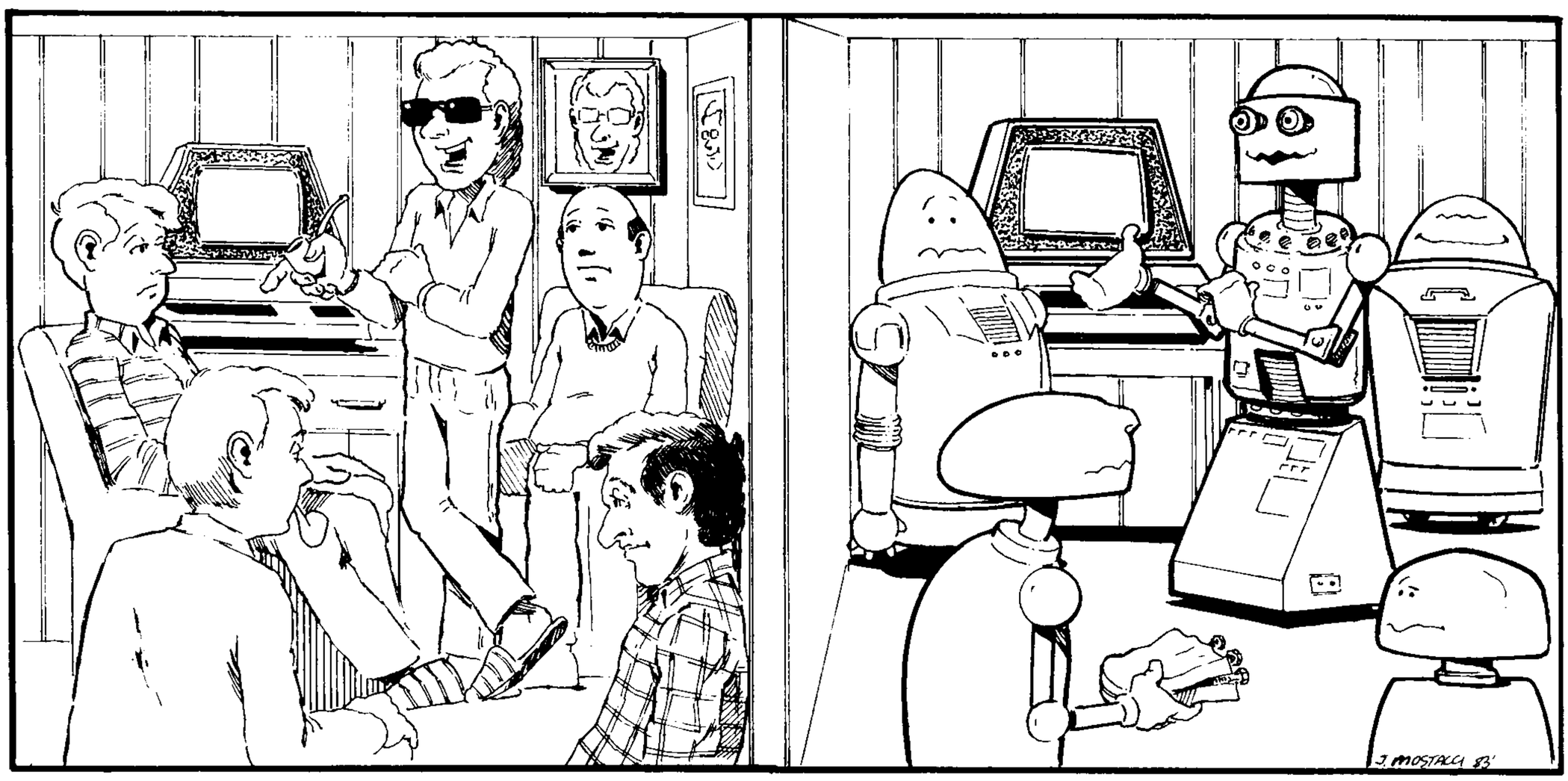
Making Friends With SID **Vol4 Iss3 Pg42.**

Line 100 should read:

100 if peek(key)<>64 then **100**

Tapemaker **Vol4 Iss3 Pg22.**

It wasn't stated clearly, but Tapemaker 4.0 only works on Basic 4.0. A VIC 20/C 64 version is under investigation.

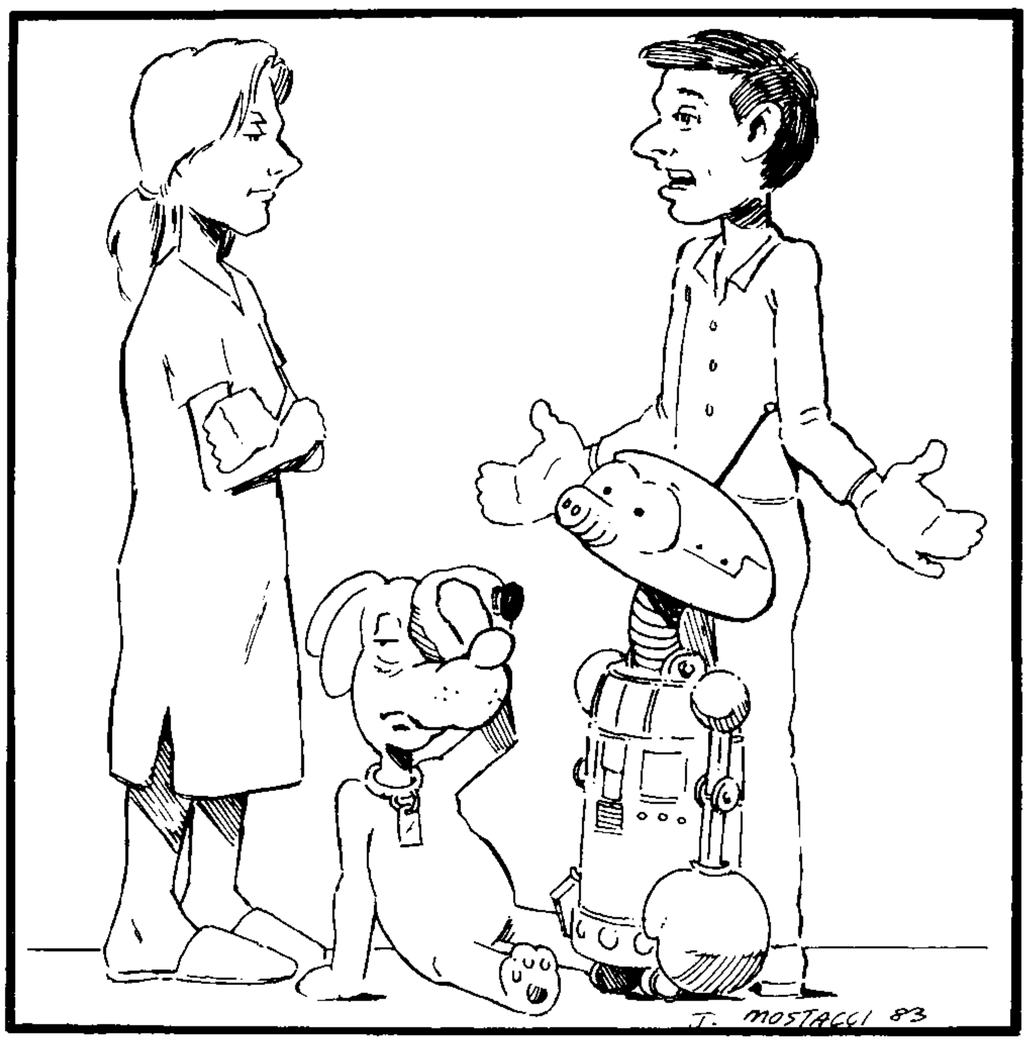


To Err Is Human, But To Really
Screw Up You Need A Computer!

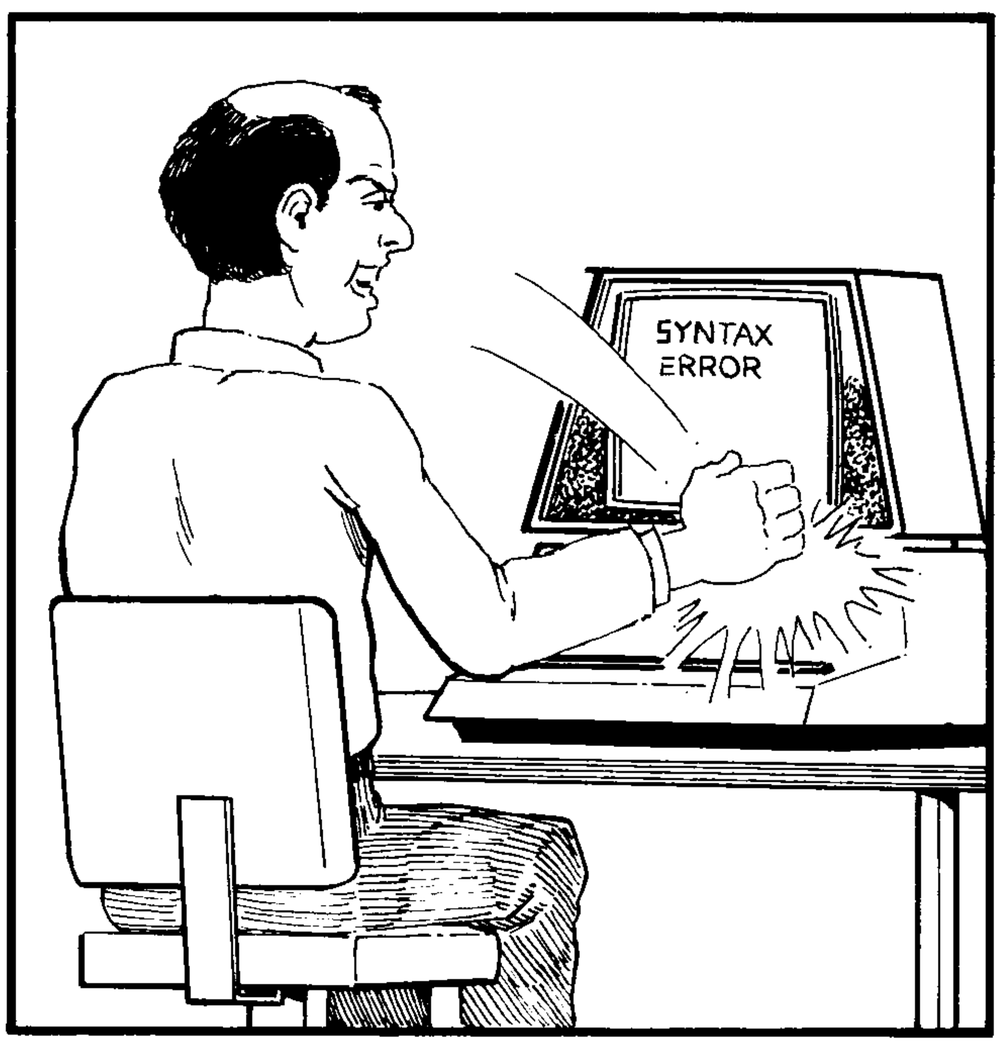
To Err Is Electronic, But To Really
Malfunction You Need A Computer Programmer

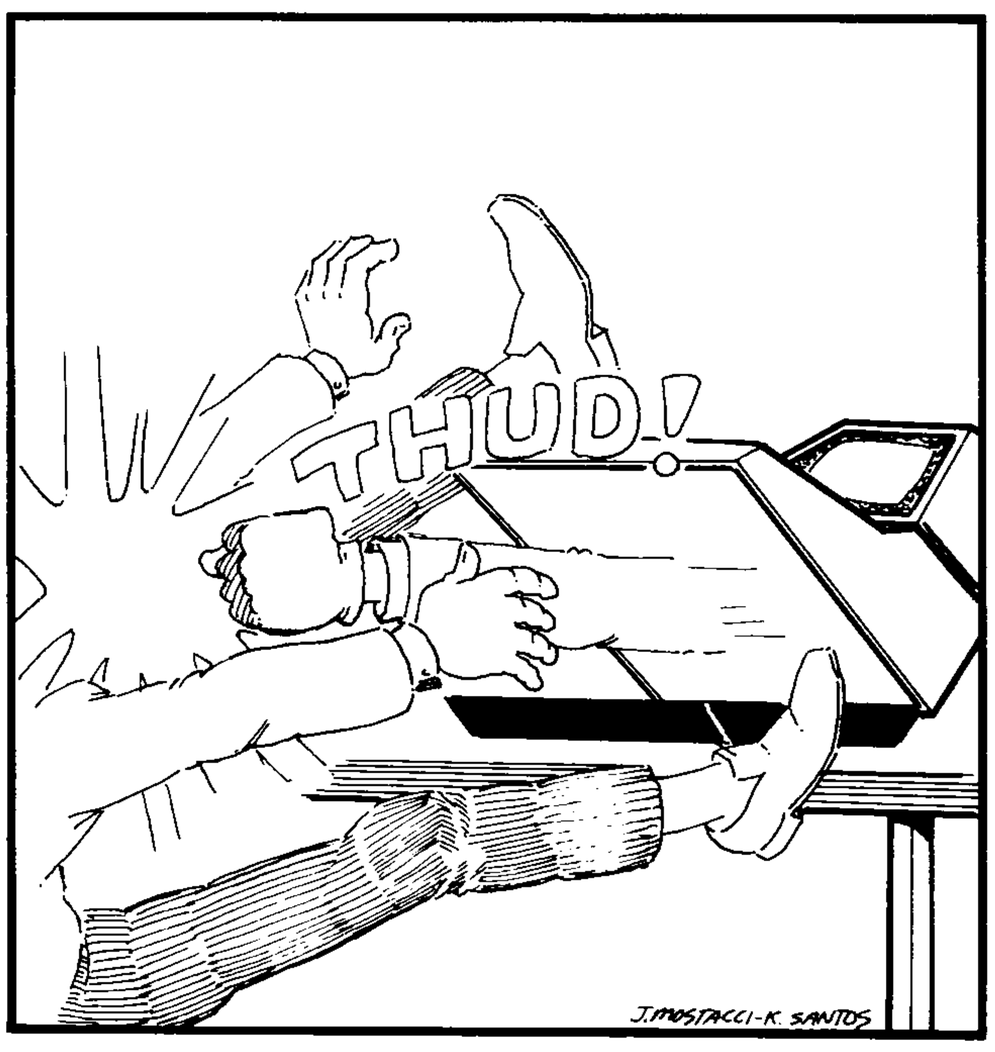
CompuKinks.

At Last We Have Someone Worth Voting For



Honest, I Only Asked T.K.O. To Put The Dog Out
And Next Thing You Know...





First Annual Toronto Computer Fair

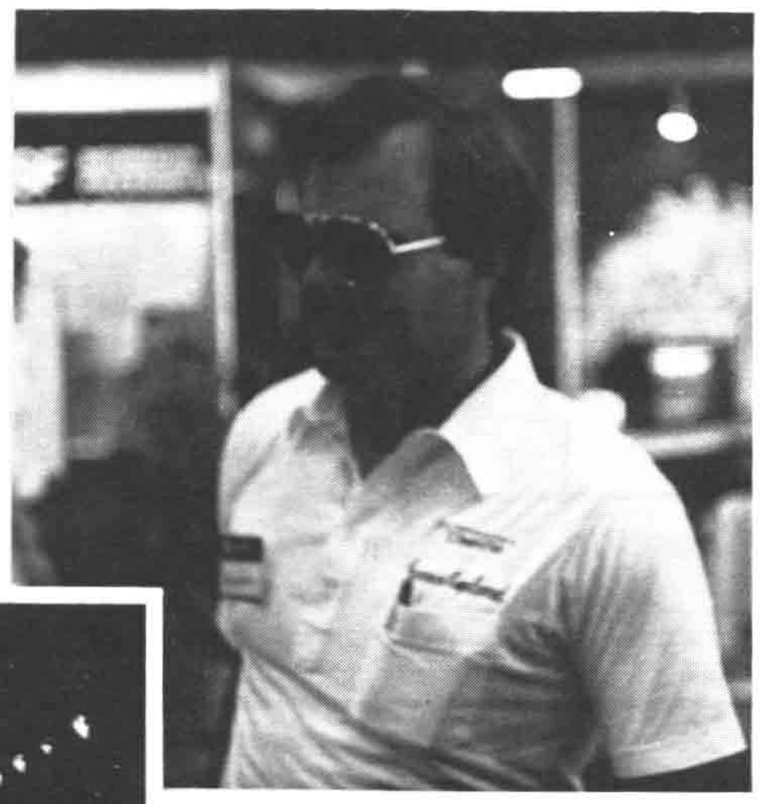
The first Computer Fair at Toronto's International Center became an instant success. Despite the heat and a perfect weekend, over 40,000 attended the summer show. Unconfirmed reports say The Fair will be moved to a date outside the summer months for next year which will no doubt be even bigger.

The exhibition was primarily microcomputers and peripherals, with a good mix of software, accessories, books, etc. Some neat synthesizer demos, the occasional droid, the club corner, and the dozens of video games, all contributed to a most satisfying variety. I thought the stereo displays were a little out of place though.

New items on the Commodore booth were mainly the Executive–64 (below) and a speech synthesizer cartridge with some good educational software for it. Commodore also had a MicronEye Bullet Camera (see News BRK) connected to a 64. The picture quality was pretty sharp and probably could be even better given proper lighting.

An event like this needs a good first off. I suspect that many vendors/manufacturers declined the event for lack of a reputation. But I'm sure the next Computer Fair won't have an empty booth in the entire building.

Dave Berezowski, Ranaan Mintz, and James Copland of Commodore Canada



The WordPro Book Of Tricks

Donna Green
Commodore Canada

History Of The WordPro Series

In the last issue we discussed WordPro 3 Plus – a new word processor for the Commodore 64. This raised questions about the other WordPro's and the differences between them all. Here we will address those questions with a view to how it all began.

Steve Punter – Creater Of The Series

There is now a WordPro program for virtually every Commodore computer, starting with WordPro 1 up to the new WordPro 6 Plus. It all started when computer enthusiast Steve Punter, at age 20 bought an original 8K Pet computer that had a built in cassette unit.

His intentions were to create a text editor for home use. At that time he was already familiar with word processing systems and wanted to create a program of his own that would have the same kind of editing functions. The WordPro series was eventually to develop out of his attempts, but not before a few preliminary editions.

Initially he wrote a simple text editor to which he later added a formatter for screen and printer output, and named it the "DCE Text Editor" (Direct Cursor Editor). This was late 1978 and there was no printer available for testing out the program further. However, the program was circulated for a while and one of the impressed recipients contacted Steve for more information. Eventually they formed a successful alliance that still continues today – several programs later.

Encouraged through the response of his programs and the formation of a new company, Steve started to rewrite this program and created "Status Line EF8" for 16K Pets. So far everything had to be hand assembled. Finally in May 1979 Steve was able to use a printer with his programs for the first time.

"WordPro 2" For The 2001

WordPro 2 evolved later in 1979 and was the first program to become commercially available. It was written specifically for the newer 16k Commodore 2001 computer and to Steve's great delight, an assembler was also now available. There was also a new disk drive (the 2040) on the scene. WordPro 2 had a menu of editing commands, including such features as Append, Load, Merge, and Save, and it offered up to 173 lines of text.

This original version ran only with Basic 2.0 and became obsolete when later, the 2001's were upgraded to Basic 4.0.

"WordPro 1" On Cassette For The 8K PET

The success of WordPro 2 drew Steve back to his earlier 8K programs and he completed revisions for the creation of WordPro 1. This package, like all WordPro programs, comes complete with a manual and a program tape or diskette. It is still available on cassette for the 8K or 2001 Pet. One side of the program tape has WordPro 1 Basic 1.0 and the other side 2 contains the Basic 2.0 version.

Its functions include a status line and many of the popular text editing functions inherent in all the WordPro's that follow. There are up to 53 lines of text per file allowed, and it features the still present "wrap around" at the end of lines.

"WordPro 3" For The 4032

WordPro 3 was designed for the 4032 computer and was far better than anything else to date. It combined all the features of WordPro 2 and added many more. A company called "Professional Software Inc." would become the distributor of these programs.

"Saving" files was called "Memorizing" for the first time. Global functions allowed for global search and replace, global repagination through linking files, global printing with automatic page numbering and headers and footers. Files could be up to 338 lines long, and an "extra text" area could contain at least 23 lines, with these numbers of lines being adjustable, between the two work spaces.

"WordPro 4" For The 8032

WordPro 4 was introduced to the business market when the new 80 column screen appeared. It was a twin to WordPro 3 except for the additional feature called "Output to Video" which was made possible by the wider screen. This allowed the operator to view the text in its formatted form before print out.

Finally – "WordPro 3 Plus" and "WordPro 4 Plus"

Eventually, WordPro 3 was replaced by WordPro 3 "Plus" – an improved edition that now includes many more printer functions such as bold face print, variable line spacing and character spacing, and the ability to add and subtract columns of numbers.

At the same time in late 1981, WordPro 4 was also obsoleted by its updated version – WordPro 4 "Plus". The same extra features of "3 Plus" have been included as well as few more, such as a directory in four columns for easier viewing, with tab saving ability. This highly sophisticated program still rates highly with the top word processors on the market today. "WordPro 4 Plus" runs on the 8096 and the SuperPet as well as the 8032.

"WordPro 2 Plus" – Word Processing For All Systems!

In 1982, an all new "WordPro 2 Plus" was created to run on any system with a minimum of 16K. Fashioned after WordPro 3 Plus, this was the program to answer the needs of education, or individuals who have any combination Commodore hardware (4016, 4032, 8032, 8096, SP9000, 4040, 8050 or cassette units), therefore it runs with either 80 column screens, "Fat 40", or regular 40 columns. The configuration is selected at the startup of the program. Basically the same as "3 Plus", it has a few less features due to the limit of 16K memory.

"WordPro 3 Plus For The 64"

WordPro 3 Plus was converted for the Commodore 64 computer completely intact, and has been available since January 1983.

"WordPro 4 Plus ML" (MultiLingual)

This is the same great WordPro 4 Plus program with a difference. The "ML" version, adapted by an associate in Montreal, will allow for text to be entered in French, German, Spanish or Italian (as well as English), with all the appropriate characters appearing on the screen. Such characters are easily accessible through the redefining of several keys and the use of a special character generator ROM. Letter quality printers must have specific type fonts to function accurately.

"WordPro 5 Plus" For The 8096

With every new computer there has emerged a new WordPro, and thus WordPro 5 Plus evolved – even though "4 Plus" will also run with the 8096. The major advantage to WordPro 5 Plus is the size of the text files and work spaces. There are 169 lines available in each of 5 separate text areas. The main text and 4 extra text areas can be likened to working on 5 different files at any time, switching from one to another for reference, or editing and updating, as required.

"WordPro 6 Plus" For The SuperPET (SP9000)

Like "5 Plus", this program also offers multiple text areas, however now the number of lines in each is somewhat definable. The operator is asked to select either "Short or

Long Banks" of text. "Short banks" will provide 6 text areas of 153 lines each, while "Long banks" will provide 3 text areas of 256 lines each as well as 2 areas of 51 lines each. This program is available as of July 1983.

Future WordPro's?

Already well underway, is a program for the Commodore 64 that is a total reconstruction of the WordPro series. This word processor, to be called "WordPro 64", is promised to be based on the most modern concepts developed in recent years, and will have many incredibly advanced features that Steve Punter feels will create the best word processor ever. Needless to say, we are all eagerly awaiting its release.

Extra Manuals

Two independent manuals have now been published to help owners and students learn WordPro programs. Both are complete with step by step instructions and practical exercises. One manual published by Gage Publishing is entitled "Introduction to WordPro 4 Plus", authored by two professors from the University of Western Ontario. The other is published by Copp Clark Pitman and authored by three teachers in the North York Board of Education. It is entitled "WordPro for Commodores, A Student Manual".

WordPro Compatible Programs

"MailPro" (By Steve Punter)

MailPro is especially versatile in setting up mailing lists for merging with WordPro files, or for printing out mailing labels or envelopes. Over 2,000 records can be stored on one disk (depending upon disk drive), and fields can be easily sorted and retrieved just like a full–fledged data base program.

"SpellPro" (By Jim Butterfield)

Just what so many have hoped for! "SpellPro" will check the spelling in WordPro files against the "Butterfield Basic" dictionary. The "unmatched" words are marked for either correction, deletion, and/or addition to the dictionary which can be extended up to 80,000 words (depending upon disk drive).

All the above programs – WP1, WP2+, WP3+, WP4+, WP5+, WP6+ and SpellPro and MailPro – are available from authorized Commodore computer dealers.
Below is a program that will allow you to read WordPro files if you do not have a WordPro program. Thats all for now, more next time.

```
100 input "filename";f$
110 input "drive #";dr
120 z$=chr$(0) : s$=chr$(32)
130 open 1,8,3,mid$(str$(dr),2)+":"+f$+",p,r"
140 get#1,a$,a$
150 get#1,a$ : ss=st
160 a=asc(a$+z$)
170 if a>64 and a<96 then a=a+128
180 if a<32 then a=a+64
190 print chr$(a); : if ss then 240
200 if a<>95 then 150
210 print
220 get#1,a$ : if a$=s$ then 220
230 goto 160
240 close 1
```

The MANAGER Column

This issue, the Manager Column is a couple of tips submitted by readers, followed by a hard disk backup program for those using a 9060/9090 drive or those considering such an addition in the future. The illustrious John Stoveken will return next issue with a list of the SYS BA+'s and their functions. – Ed.

Manager Searching

Ross Eddie
West Hill, Ont.

To add a "Hunt" like search function to Global Update, Produce Sub–Files (filename "finish" on the diskette), and Report Generate, modify lines 800 to 850, and 2356. Then add lines 2702, 2704, 2706, and 2708 as follows:

```
800 sys pr, 24, 0, "enter condition 1.< 2.> 3.<= 4.>=
    5.= 6.<> 7. any occurence ?"
810 a$=str$(sc%(i) : sys pr, y, x3–1, a$
820 x$="" : sys in, y, x3, 1 : gosub 3345 : sc%(i)=val(x$) :
    if sc%(i)<1 or sc%(i)>7 then 820
850 sys pr, 24, 0, "[space61]"
860 sys pr, 24, 0, "enter search string?"

2356 on sc%(j) gosub 2420, 2470, 2520, 2570, 2620, 2670,
    2702 : goto 2370
2700 return
2702 ae$=right$(ss$(j),(len(ss$(j))–1))
2704 for jj=1 to (len(r$)–len(ae$))
2706 if mid$(r$, jj, len(ae$)) = ae$ then fl=1
2708 next jj : return
2710 az=ra : ra=val(t1$) : a=peek(198) : b=peek(216) :
    c=peek(ba+78+13)
```

This adds a new seventh condition to the standard six which appear in the three programs mentioned above. These are the only three that allow for search criteria, and if you want the new condition in all of them, you'll need to make the changes in each. Specifying a search condition 7 will find any occurence of the search string in any field, anywhere in the record. It also picks up spurious matches which might consist of the end of one field and the beginning of the next, but that won't bother most applications. Be extra careful with consecutive numeric fields which are most succeptable to this.

Ross's mods may be a little slow in Basic under extensive use. When Mr. Stoveken explains the SYS functions next time, they might be easily converted to machine language calls.

Device Numbers

Rev. Dave Sproule
Etobicoke, Ont.

Since I frequently have both my CBM 2022 printer (device 4) and Olivetti ET221 typewriter (device 5) fired up at the same time, I changed a line of Report Generate to read:

```
10020 poke 32767, 12 : if x$ = "A" then poke 32767, 14
      : pt = 5 : dv = pt
```

With the addition, the ASCII/CBM question also selects the correct device number.

I have come up with a few more helpful goodies but I don't know if you plan to include this type of user modification data. I for one, though, would certainly be interested in reading what other Manager users have done to improve a very good program for their particular needs.

I also have a question. In adapting The Manager by simplifying it for office use, I would like to have several programs call the next in sequence. For instance: In using "generate" to print our parish list, I ask if new names have been added since the last printing. If the answer is "Yes" then "sort" is called and run to set up a new pointer file. Unfortunately, when "sort" calls "generate" to get back to printing the list, my 8032 craches with the backslash x, "generate" command. Any suggestions?

Also, with regard to Bill MacLean's article, "File Chain Tracer" in issue 01. I think this is what The Manager has needed. I really could use Manager relative files with records about three blocks in length. How about an article on the necessary mods to do it?

First Dave, we would be delighted to print any more mods that might assist other Manager users. If anyone else has any, please write with those too. Your first question is on the investigations list and your second is indeed challenging. We'll try getting both into a future release.

Hard Disk Backup

This program, designed by James Whitewood of Milton, Ontario, takes a relative file created by The Manager from a Commodore hard disk (eg. 9060/9090) and copies it record by record to as many floppy diskettes as necessary. The program to get it back off the floppies will be printed next issue.

Procedure:

1. Enter the program (next page) and save it with the filename "HARD BACKUP". (Note: save it once on the hard disk and again on your Manager system diskette)

2. Make sure that the hard disk drive and the floppy drive have different unit addresses. This can be done with the program called "change disk" provided on The Manager system diskette. We recommend that the hard disk be defined as unit 8 and the floppy as unit 9.

3. Load "HARD BACKUP"

4. Hard Backup uses Manager machine language subroutines. Most likely you have copied the entire Manager system diskette onto your hard disk. Continue with RUN and Return and HARD BACKUP will load the necessary routines.

5. Enter the source file name.

6. Enter the source unit number (ie. the number of the hard disk).

7. Place destination diskette in drive 0 of unit 9 (ie. a diskette that will receive the source file).

8. Enter destination unit number.

9. Line 230 allows formatting of the destination diskette. Type "Y" or "N" and hit Return.

10. When the floppy has been filled, the bell will sound, and the computer will prompt with: "is another backup diskette ready?". Place another diskette in drive 0, type "Y" and hit Return. You will then be placed back at Step 9.

11. When the entire hard disk file has been copied, the computer will return to "The Manager" menu.

```
100 if fl=1 then 130
110 if fl=2 then 260
120 fl=1 : dload "basc"
130 sys 5*4096 + 8*256
140 \x;"0:cmach 1" : ba=18432
150 print "■S■";
160 for i=1 to 8 : print "**********"; : next
170 print tab(29);"backup a hard disk file"
180 for i=1 to 8 : print "**********"; : next
190 print : print "■q■enter source file name? ";
200 n=12 : gosub 460 : sf$=x$
210 print : print "■q■enter source unit number? ";
220 n= 2 : gosub 460 : su=val(x$) : if su<8 or su>15 then 220
230 print : print "■q■enter destination unit number? ";
240 n= 2 : gosub 460 : du=val(x$) : ifdu<8ordu>15orsu=duthen240
250 fl=2 : dload(sf$+".scr"), d0, u(su)
260 sys ba : nr=peek(27635)+peek(27636)*256 :rl=peek(27634)
270 y1=peek(216) : x1=peek(198):dopen#1,(sf$),d0,u(su):dn=0
280 print : print "■q■do you wish to format the destination diskette? ";
290 n= 1 : gosub 460 : if x$<>"y" and x$<>"n" then 290
300 if x$="y" then gosub 540
310 d$="0" : gosub 580
320 tr=int(fb*254/(rl+1)) : if tr=0 then 390
330 dn=dn+1 : dopen#2,("@"+sf$+str$(dn)), d0, u(du), w
340 if dn=1 then dopen#3,"file data", d0, u(du), w : print#3,nr,rl:dclose#3
350 for i=1 to tr : x$=""
360 \i,1,x$ : x$=x$ : if x$=chr$(255) then dclose#1 : dclose#2 : goto 440
370 me$="v"+str$(i) : gosub 510
380 print#2,x$ : nexti : dclose#2 : me$="v" : gosub 510
390 print : print "■qqqqq■is another backup diskette ready? ";
400 n=1 : gosub 460 : if x$<>"y" then 400
410 poke 216, y1 : poke 198, x1 : sys 57447
420 for i=1 to 5 : print"■v■" : nexti
430 poke 216, y1 : poke 198, x1 : sys 57447 : goto 280
440 me$="finished backing up the file" : gosub 510
450 \x,"0:the manager"
460 y=peek(216) : x=peek(198)
470 x$="" : sys ba+57, y, x, n
480 if x$="Q" then run
490 if x$="q" then: \x,"0:the manager"
500 return
510 yx=peek(216) : xx=peek(198)
520 me$=me$+"" : sys ba+60, 24, 0, me$
530 poke 216, yx : poke 198, xx : sys 57447 : return
540 me$="formatting diskette" : gosub 510
550 a$="n0:"+sf$+","+right$("  "+str$(dn),2) : open 15, (du), 15, a$:close15
560 me$="v" : gosub 510
570 return
580 x$="$"+d$+":#$%&'" : open 3, (du), 0, x$ : get#3,x$,x$,x$,x$,x$,x$
590 get#3,x$ : if len(x$)<>0 goto 590
600 get#3,x$,x$,x$,y$ : x=len(x$) : if x then x=asc(x$)
610 y=len(y$) : if y then y=asc(y$)
620 fb=x+y*256-5 : close 3 : return
```

Superkey–64

Darren Spruyt
Gravenhurst, Ont.

This is an adaption of a keyword utility, by Charles Brannon, for the Commodre–64. Burst keys are a handy thing to have around, as it shortens the time needed to enter keywords such as LIST, RESTORE, GOSUB and RETURN. Other keyword programs allow only 26 keywords to be included in their table, while mine allows up to 52 and 2 specials.

This keyword program works off of the hardware interrupt vector located at $0314–$0315 (the $ denotes hexadecimal notation, in decimal the values are 788–789). The program re–routes this vector to go to the machine language routine located at $c000 – $c190.

At $c000 the routine checks to see if the Function 1 key has been pressed, if it has, it goes into waiting mode. The program will stay in waiting until another key has been pressed. The legitimate keys to press are A through Z and shifted A through shifted Z. If any other key has been pressed, the program justs exits from the waiting mode. If a legitimate key has been pressed, the program looks up the token for that key in a table, and then it prints the keyword that the token represents.

Tokens

A token is a single byte used to represent a keyword. This way, a whole keyword such as RESTORE can be condensed into one character. Token values are always equal to or greater than 128 decimal or $80 hexadecimal. This tokenization saves a lot of memory in BASIC storage of programs. Upon LIST, BASIC automatically expands these tokens into the keywords that we are all familiar with.

Table of Tokens and the keywords that they represent:

128 – end	129 – for	130 – next
131 – data	132 – input#	133 – input
134 – dim	135 – read	136 – let
137 – goto	138 – run	139 – if
140 – restore	141 – gosub	142 – return
143 – rem	144 – stop	145 – on
146 – wait	147 – load	148 – save
149 – verify	150 – def	151 – poke
152 – print#	153 – print	154 – cont
155 – list	156 – clr	157 – cmd
158 – sys	159 – open	160 – close
161 – get	162 – new	163 – tab(
164 – to	165 – fn	166 – spc(
167 – then	168 – not	169 – step
170 – +	171 – –	172 – *
173 – /	174 – ↑	175 – and
176 – or	177 – >	178 – =
179 – <	180 – sgn	181 – int
182 – abs	183 – usr	184 – fre
185 – pos	186 – sqr	187 – rnd
188 – log	189 – exp	190 – cos
191 – sin	192 – tan	193 – atn
194 – peek	195 – len	196 – str$
197 – val	198 – asc	199 – chr$
200 – left$	201 – right$	202 – mid$
203 – go		

Notice that some tokens represent single characters such as +, >, and ±. When these characters are used outside quotes, BASIC assumes they will be part of some arithmetic operation. Within qoutes, the same characters have a different value which is a PET ASCII code. By assigning two different values to the same character – depending on where it is used – BASIC can more quickly and easily determine the difference between a "+" meant for display and a "+" meant for execution.

Example of Tokenization

Type the following as it appears

```
new
10 rem
poke 2053, 128
list
Now the line that should appear is
10 end
```

The value in location 2053 is the token that controlled the keyword that BASIC displayed just after it printed the line number. By changing the value in location 2053, according to our table above, we can control which keyword BASIC will display.

Now lets go on to application of the program. In the first paragraph, I mentioned that the program could take up to 52 keywords in its table and two specials. These specials are one-key entries of two commands, LIST and RUN. By pressing the F5 key, the BASIC program in memory is automatically LISTed from the beginning. No line range can be set such as LIST 100–200 with this function. The F7 key is an auto run key, just press it and the current BASIC program in memory is automaticaly run. As with the previous command, no line can be set for the run to begin at.

The other keywords are displayed using the above description of the waiting mode. When the F1 key is pressed, the computer is set up to wait for another key entry. After this, if any key is pressed,©it decodes it and prints a keyword, if neccessary, otherwise it just exits from waiting.

An Example

Press the F1 key and then press the A key. The computer should now display the keyword POKE where the letter A used to be. Now for another. Press the F1 key and press the shift key and the A key together like you were going to get the graphic character off of the A key. The keyword PEEK should now appear where the graphic A character was.

The F3 key is also useful as a special key which has already been defined, this is to be used as a delete key. If, for instance, a wrong keyword has been displayed on the screen which is four characters is length, the delete key would have to be pressed four times. With this key, you just push it and the keyword just displayed, would be deleted. This key is just primarily useful to delete incorrectly entered keywords.

The keywords that are displayed are in accordance with the keyword table DATA statements that appear at the end of the program, namely lines 135–139. The table ascends for A to Z then from shifted A through shifted Z. If, for instance, the first data number in line 135 is changed from 151 to 176 and the program run again the following will happen. Upon pressing the F1 key and then the A key, the keyword OR will be placed on the screen rather than the keyword POKE.

Another method of changing the keyword that will be displayed, after the program has been lost from memory is to modify the table in memory. At $c120 or 49440 in decimal is a table identical to that of the last data statements. By POKEing to this table, we can also change the keywords that are displayed. The table is also set up as in the data statements. Press the F1 key and then the A key; the keyword OR or POKE should be displayed. Now POKE 49440,128 and do the same procedure. The keyword END should now have been displayed.

Another item to remember is how to turn the program on and off. To enable, use the SYS that is given at the beginning of the program. To disable, press RUN-STOP/RESTORE.

By modifing the keywords that are displayed by using the table and/or the data statements, the keys and their matching keywords can be set up in a way that can be used easily and efficiently by the user.

Editor's Note

Great program Darren! Hope to see more like it! One thing I noticed, that might be considered an added bonus, is the fact that your routine doesn't affect the value of the F1 key from a GET statement. Too often this is overlooked. Also, the utility is still in affect during an INPUT statement, but again with no nasty side effects.

As of running the program, the setup of the keys is as follows:

Un-Shifted	**Shifted**
a = poke	A = peek
b = clr	B = chr$
c = print#	C = cont
d = data	D = dim
e = end	E = def
f = for	F = abs
g = goto	G = gosub
h = get	H = return
i = input	I = input#
j = mid$	J = stop
k = right$	K = atn
l = left$	L = rem
m = tab(	M = exp
n = next	N = not
o = log	O = open
p = print	P = close
q = rnd	Q = cmd
r = read	R = restore
s = step	S = str$
t = then	T = asc
u = save	U = fre
v = load	V = wait
w = verify	W = int
x = len	X = spc(
y = val	Y = sqr
z = sys	Z = rnd

```
10 print " Sqq superkey 64 "
20 print " qq by
30 print " qq darren spruyt "
40 print " qqqqq please wait  "
50 for j=0 to 339 : read x : poke 49152+j, x
60 poke 1024, x : ch=ch+x : next
65 if ch <> 48346 then print " data error "  : stop
70 print " Sqqqqq to turn on: 'sys 49409' "
80 print " qq to turn off 'run stop/restore' "
90 print " qq table at 49440 "
99 end
100 data 142,  18, 193, 140,  19, 193
110 data 141,  16, 193, 173,  17, 193
120 data 240,  77, 165, 215, 201, 133
130 data 208,  11, 169,   0, 141,  17
140 data 193, 234, 234, 234,  76, 244
150 data 192, 201, 134, 208,  29, 169
160 data   0, 133, 198, 174,  20, 193
170 data 240, 237, 169,  20, 230, 198
180 data 164, 198, 153, 119,   2, 200
190 data 202, 208, 245, 160,   0, 234
200 data  76, 212, 192, 234, 201, 135
210 data 208,   4, 169, 155, 208,   6
220 data 201, 136, 208, 205, 169, 138
230 data 160,   1, 140,  23, 193, 238
240 data  22, 193, 208,  44, 234, 234
250 data 234, 165, 215, 201, 133, 240
260 data 184, 201, 193, 144,  10, 201
270 data 219, 176,   6,  56, 233, 167
280 data  76, 122, 192, 201,  65, 144
290 data 163, 201,  91, 176, 159,  56
300 data 233,  65, 160,   0, 140,  23
310 data 193, 234, 170, 189,  32, 193
320 data 170, 160,   0, 132, 198, 160
330 data 158, 132,  34, 160, 160, 132
340 data  35, 160,   0,  10, 240,  17
350 data 202,  16,  16, 230,  34, 208
360 data   2, 230,  35, 177,  34,  16
370 data 246,  48, 241, 234, 234, 198
380 data  34, 200, 177,  34, 234,  48
390 data  10, 230, 198, 166, 198, 157
400 data 119,   2,  76, 169, 192, 230
410 data 198, 166, 198,  41, 127, 157
420 data 119,   2, 173,  23, 193, 240
430 data  12, 169, 138, 141, 119,   2
440 data 141,  23, 193, 208,   8, 234
450 data 234, 136, 169,  20, 141, 119
460 data   2, 230, 198, 141,  17, 193
470 data 140,  20, 193, 173,  22, 193
480 data 240,  14, 166, 198, 169,  13
490 data 157, 119,   2, 230, 198, 169
500 data   0, 141,  22, 193, 173,  16
510 data 193, 172,  19, 193, 174,  18
520 data 193,  76,  49, 234, 234, 120
530 data 169,   0, 141,  20,   3, 169
540 data 192, 141,  21,   3,  88,  96
550 data  96, 234,   0, 138, 236,   3
560 data   4, 234,   0, 138, 234, 234
570 data 234, 234, 234, 234, 234, 234
580 rem **** keyword token table ****
590 data 151, 156, 152, 131, 128, 129
600 data 137, 161, 133, 202, 201, 200
610 data 163, 130, 188, 153, 187, 135
620 data 169, 167, 148, 147, 149, 195
630 data 197, 158, 194, 199, 154, 134
640 data 150, 182, 141, 142, 132, 144
650 data 193, 143, 189, 168, 159, 160
660 data 157, 140, 196, 198, 184, 146
670 data 181, 166, 186, 186
```

Re-Decoding The 8032 Keyboard

Greg Beaumont
Ingersoll, Ont.

This program demonstrates the use of output jump vector at decimal 235 ($eb) on the CBM 8032. The program accomplishes it by catching the keys as they are pressed and using the ASCII value of the key pressed as an index into a table. The value at that location in the table is stored at location 217 (last key pressed) and the regular subroutine in ROM is called.

The sample program included remaps the shift of the numeric pad to some of the extended screen functions available on the 8032,8096 and SuperPet. However, these could easily be changed to other characters such as graphics which are not readily available from the keyboard.

```
10 poke 53, 126 : clr : rem lower top of memory
20 print " SRE-DECODE 8032 V1.0 - GREG BEAUMONT "
24 print
25 print " This utility redecodes the keyboard on the CBM 8032. "
26 print " Below is a list of the new key functions "
27 print " To use them press then 'SHIFT' key and the number of the desired "
28 print " function on the numeric key pad "
29 print
30 print " SHIFT '1' - ERASE BEGIN "
40 print " SHIFT '2' - ERASE END "
50 print " SHIFT '3' - BELL "
60 print " SHIFT '4' - SCROLL DOWN "
70 print " SHIFT '5' - TEXT MODE "
80 print " SHIFT '6' - DELETE LINE "
90 print " SHIFT '7' - SCROLL UP "
95 print " SHIFT '8' - GRAPHICS MODE "
96 print " SHIFT '9' - INSERT LINE "
100 gosub 1000
110 sys 7*4096 + 14*256 : end
1000 for j=32256 to 32312 : read x : poke j,x : next : return
1010 read x : ch=ch+x
1020 poke j, x : next
1030 if ch<>7006 then print " data error " : stop
1040 return
1050 data 169,  16, 133, 235, 169, 126, 133, 236
1060 data  96, 170, 170, 170, 170, 170, 170, 170
1070 data 165, 152, 240,  25, 165, 217,  10,  74
1080 data 233,  48,  48,   9,  72, 233,  57,  16
1090 data   4, 104, 168, 185,  48, 126, 133, 217
1100 data 160,   0, 136, 208, 253,  76,  12, 226
1110 rem the next 9 are the table of substitute values
1120 data 150,  22,   7, 153,  14,  21,  25, 142, 149
```

Source Code

```
;*********************************************
;*        re-decode keyboard for ascii       *
;*        and new control codes by           *
;*        greg beaumont                      *
;*        keys are set to the following list  *
;*        1 - erase begin                    *
;*        2 - erase end                      *
;*        3 - bell                           *
;*        4 - scroll down                    *
;*        5 - text                           *
;*        6 - delete line                    *
;*        7 - scroll up                      *
;*        8 - graphic                        *
;*        9 - insert line                    *
;*********************************************
;
;
  *     = $7e00
lstkey  = $d9
shftky  = $98
start   = $7e10
init    lda   #<start  ; link redecode data
        sta   $eb
        lda   #>start
        sta   $ec
        rts
  *     = start
        lda   $98      ;get status of shift key
        beq   exit     ;check for shift key pressed otherwise exit
        lda   lstkey   ;get last key pressed
        asl   a        ;mask off bit 7
        lsr   a
        sbc   #'0      ;ascii $30
        bmi   exit     ;must be between 1 and 9
        pha            ;save old value
        sbc   #'9      ;ascii $39
        bpl   exit
        pla            ;restore old value
        tay
        lda   table,y
        sta   lstkey
delay   ldy   #$00
        dey
        bne   delay
exit    jmp   $e20c
table ;
        .byte
        150,22,7,153,14,21,25,142,149
        .end
```

Making Friends With SID Part 3

Paul Higginbottom
Toronto, Ont.

Yes folks, it's that time again. Hopefully by now, you will be starting to see the various interactions of envelopes, frequencies, and waveforms. These are our basic tools for music (or indeed any sound) synthesis.

To recap, the program shown last article calculates a frequency array of 7 octaves of notes. We will use that as a base for this article, to create a "piano" program, which will allow us to play the keyboard like a piano.

However, before we begin adding lines to the program, you will need to delete all the lines from 450 onward (by simply entering the line numbers by themselves and pressing RETURN), because we certainly don't want to see, or hear(!) the frequency table being created every time we run the program.

To allow us to play the keyboard, our program must see if a key is pressed, and if it is, we must evaluate which note that corresponds to, and gate a voice on with that particular frequency. To sound "pianolike," that is, when a second key is hit (after the first is released) the first sound will still be able to be heard, our program must "cycle" through the voices. To put this another way, the first key pressed will "play" the first voice, and then the second key, the second voice, the third, the third voice, and the fourth, back to the first voice. A statement such as:

```
voice=voice+1 : if voice>2 then voice=0
```

. . .will cycle the voice counter; the IF statement ensuring the variable "VOICE" doesn't go out of the range 0 to 2.

Enter the following lines after having loaded the program from the last article, and having deleted lines 450 onward:

```
500 sid=54272
510 voice=0 : oct=4 : wave=32
520 vm=7 : hi=256
530 for i=0 to 23 : poke sid+i, 0 : next
540 poke sid+24, 15
550 read a, d, s, r, pw
560 for i=0 to 2 : index=sid+i*vm
570 poke index+5, a*16+d
580 poke index+6, s*16+r
585 poke index+2, pw and 255
586 poke index+3, pw/hi
590 next
600 data 5, 11, 0, 0, 800
700 k$=" q2w3er5t6y7ui9o0p@-  "
710 dim k(255)
720 for i=1 to len(k$)
730 k(asc(mid$(k$,i,1)))=i
740 next : print
750 print "  2 3   5 6 7   9 0   -   "
760 print " q w e r t y u i o p @*  "
800 get a$ : if a$="" goto 800
810 key=k(asc(a$))-1 : if key<0 goto 800
820 if key>11 then frq=f(oct+1,key-12) : goto 835
830 frq=f(oct,key)
835 fh=int(frq/hi) : fl=frq-fh*hi
840 index=sid+voice*vm
850 poke index, fl : poke index+1, fh
860 poke index+4, wave : poke index+4, wave+1
870 voice=voice+1 : if voice>2 then voice=0
880 goto 800
```

Line by Line Explanation

500 assigns the address of the SID chip to the variable "SID".
510 initializes variables "VOICE" (current voice being 'played'), "OCT" (current octave the keyboard notes begin at), and "WAVE" (the current waveform value to be POKE'd into the control register of a voice).
520 sets two constants; "VM" (voice multiplier) to 7 because there are 7 bytes to control each voice, and the memory address of the start of each voice is computed by "VOICE*VM", and "HI" to 256, which is the high order value divisor. It was explained in a previous article how certain values are represented in two values; "lots" of 256, plus the remainder.
530 this FOR. . .NEXT loop initializes the SID chip registers to zero, to ensure that no previously run program will affect this one.
540 set SID register 24 to 15. This is the volume register, and therefore we are setting the volume, and 15 is the maximum value.
550 reads in the envelope and pulse width parameters we wish to test from the DATA statement at line 600. The variables A, D, S, and R represent the four parts of the envelope; Attack, Decay, Sustain, and Release. PW is the Pulse Width.
560 start of a FOR. . .NEXT loop which will set the three voices in the SID to the parameters just READ. "INDEX" is set to the start address of the block of seven bytes for the current voice.
570 put the attack and decay values into register 5 of the current voice.
580 put the sustain and release values into register 6 of the current voice.
585 put the low value of the pulse width into to register 2 of the current voice.
586 put the high value of the pulse width into to register 3 of the current voice.
590 continue looping until done with the NEXT command.
600 DATA to be tested. This consists of ATTACK, DECAY, SUSTAIN, RELEASE, and PULSEWIDTH (if that waveform is used).
700 k$ is set to hold all of the keys which can be played. You should note that these keys represent a piano layout on the keyboard.
710 dimension the K(array. This will be an array of the key positions, subscripted by the ASCII value of the keypress. This is necessary, because we need to know the key "position" along the piano layout to calculate the frequency which should be used. Therefore, k(asc("q"))=1, k(asc("2"))=2, etc.
720 start of FOR. . .NEXT loop to assign the K(array. The loop goes from 1, the number of characters in k$, which is calculated using the LEN(function.
730 the expression asc(mid$(k$,i,1)) gives the ASCII value of the i'th character in k$.
740 continue the loop.
750–760 print up the keyboard on the screen.
800 *main loop* – check keyboard, if nothing is pressed, go back to this line again, and continue checking.
810 assigns KEY to the position of the keypress on the keyboard minus one. Since elements of the array for keys not in the piano part will be equal to zero, this expression will equal minus one if an invalid key is pressed. Therefore, the IF statement checks for invalid keys by seeing if KEY<0, and if so, going back to the keyboard checking line again.
820 there are 12 keys/notes within an octave. Since there are more than 12 keys in our piano layout, the program must check for this, and if the key position is greater than 11 (since the key position goes from zero onward), it must subtract 12 from the key (since key is the semitone value) and add 1 to the octave. Instead of ACTUALLY adding 1 to the octave, and subtracting 12 from the key this line evaluates the frequency right there with the subscripts oct+1, and key–12) and skips the next line which would calculate the frequency normally.
830 gets frequency from the array.
835 evaluates the low and high values of the given frequency in variables FL, and FH.
840 INDEX is set to the start of the vioce block of registers.
850 the frequency low and high values are now POKE'd into the SID. INDEX is equal to the position of the low byte of the frequency, and INDEX+1 to the high byte.
860 because we cannot detect the duration of the keypress, the voice is gate'd off, and then on, so that the A–D–S cycle is what is executed when a key is depressed. The voice is gated OFF first so that the release will have been completed before we gate the voice ON. Register 4 (the control register) is POKE'd with WAVE (the value of the current waveform, one of 16, 32, or 64 [128 for noise, but this isn't particularly 'musical']) to release, and then WAVE+1 to start the cycle.
870 this "cycles" the voice, as was explained earlier, so that each successive keypress uses the 'next available' voice, and this gives the use the ability to more or less play chords by playing keys in rapid succession.
880 this loops the program back to the start of the main loop to check the keyboard again.

Read the description of the program carefully, until you understand how it works. Check for any mistakes, and SAVE the program BEFORE entering the RUN command, because if you've made a mistake, you might just cause an unrecoverable crash, and have to do all that typing again!

When you've SAVE'd the program, run the program and you SHOULD (famous last words!) after a second or two, see the layout of the keys to play appear on the screen. You can then play the keys, and you'll hear the envelope which the program is using.

Ok mistro, the playing's over, it's back to work. Now it's time to experiment! The key line to change is the DATA statement in line 600. Press the STOP key to end the program, and enter:

```
600 data 3, 9, 0, 0, 800
```

This will make the attack and decay shorter than before. You may note we have a sustain level of zero, which means the decay will fade to NO volume (or very close).

RUN the program again and hear the difference. Experiment with other numbers. For example, if you have the sustain (the third number) greater than zero, the notes will 'stay on' until the VOICE variable has cycled around to the same voice again and it is changed to the new note.

The VOICE cycling line, i.e., 870, could be changed so that the piano becomes "monophonic" (one voice) by changing the "voice>2" to "voice>0". Try that. Also, the octave and waveform could be changed. Try entering:

```
510 voice=0 : oct=2 : wave=64
```

This will produce notes two octaves lower than before, and is now using the "reedy" pulse wave.

This simple program actually provides quite a bit of flexibility, although because it's written in BASIC, it is limited in that no 'real time special effects' can be performed, and the duration of keypresses cannot be checked. Look out for a cartridge coming out soon which does all this and more! Also, a real synthesizer keyboard add-on with more SID chips is coming soon!

I would expect this is enough for this time. This program should give you a much better "feel" for how the various parameters affect the sound. Next time we can go on to look at the filter. Have fun!

Sound Help

Darren Spruyt
Gravenhurst, Ont.

Sound Help is a fairly useful utility that can help with sound on the Commodore 64. With it, the user just has to call an SYS with five parameters after it to activate. Most of the information needed to use Sound Help is contained along with the program. After running the program, it explains the parameters that must follow the SYS. Have fun with it.

Editor's Note

The "lo freq value" and "hi freq value" are the low and high bytes that make up the frequency to be played. On page 384 of the C64 Programmers Reference Guide is a chart that lists all the combinations.

Sound Help Demo Program

```
1900 a=1 : e=32 : oc=4
1905 poke 54272+24, 15 : rem turn volume on
1910 read b, c, d, f : if b<0 then run
1920 sys 49664, a, b/oc, c/oc, d, e
1930 for j=0 to 110*f : next
1940 sys 49664, a, 0, 0, 0, e : rem turn voice off
1950 for j=0 to 10 : next
1960 goto 1910
2000 data  91,  50, 255, 3
2010 data  91,  50, 255, 2
2020 data  91,  50, 255, 1
2030 data  91,  50, 255, 3
2040 data 106,  63, 255, 1
2050 data 221,  59, 255, 2
2060 data  91,  50, 255, 1
2070 data  91,  50, 255, 2
2080 data 137,  47, 255, 1
2090 data  91,  50, 255, 3
2100 data  -1,   0,   0, 0
```

```
10 for d=0 to 119 : read s : poke 49664+d,s : next
15 print "[redacted]sound helper 64"
20 print "[redacted]by darren spruyt"
25 print "[redacted]to use: sys 49664,a,b,c,d,e"
35 print "[redacted]where a = voice number (1 or 2 or 3)"
40 print tab(7)"b = lo freq value
45 print tab(7)"c = hi freq value
50 print tab(7)"d = sustain/release value
55 print tab(7)"e = waveform value (16-triangle)
60 print tab(26)"(32-sawtooth)"
65 print tab(26)"(64-pulse)"
70 print tab(26)"(128-noise)"
75 print "note:"
76 print "attack/decay rate of zero is assumed."
77 print "routine not relocatable without mods."
78 print "volume to be turned on separately."
79 print "if using pulse, pulse rate must be"
80 print "set up before hand."
100 data  32,  96, 194,  56, 233,   1
110 data 201,   3, 144,   3,  76,   8
120 data 175, 141, 119, 194,  10,  10
130 data  10,  56, 237, 119, 194, 170
140 data 142, 119, 194, 169,   0, 157
150 data   5, 212, 157,   6, 212, 169
160 data   8, 157,   4, 212, 169,   0
170 data 157,   4, 212,  32,  96, 194
180 data 157,   0, 212,  32,  96, 194
190 data 157,   1, 212, 169,   0, 157
200 data   5, 212,  32,  96, 194, 157
210 data   6, 212,  32,  96, 194,  24
220 data 105,   1, 157,   4, 212, 168
230 data 165, 162,  24, 105,   3, 197
240 data 162, 208, 252, 152,  56, 233
250 data   1, 157,   4, 212,  96,  96
260 data  32, 253, 174,  32, 158, 173
270 data  32, 247, 183, 165,  21, 208
280 data   6, 165,  20, 174, 119, 194
290 data  96,  76,   8, 175, 234,   0
```

Sprite Palette For The Commodore 64

Paul Higginbottom
Toronto, Ont.

If you have a Commodore 64, chances are you've been playing with sprites. Sprites are probably one of the most popular features in a micro right now, but making a sprite is somewhat less entertaining than making it do something.

When you boil it right down, a sprite is merely 63 bytes of data arranged in such a way as to give a pattern of colours on your screen at some predetermined spot. So the first step is to arrange for this pattern. Several sprite editors are either for sale or have been published, but most force you to make sprites in a way that the computer understands more easily than you. Lining up combinations of 1's and 0's to represent the pattern of your sprite is ideal from the computers' point of view since this is exactly how the 64 will need it. Sprite Palette allows you to create sprites using blocks of colour, a far more natural approach for us humans.

The program uses multi-colour mode only. In high res mode only 2 colours are available in a sprite. Although the resolution is better in high-res, multi-colour sprites are a lot more attractive in good games, and the resolution difference is hardly noticeable.

As a sprite is being created, the actual sprite is displayed just to the right of the palette. Each sprite resides in a "page" as referred to in the instructions. Essentially each page is one palette. When a palette is used, the sprite will be displayed. The commands for moving a sprite will allow you to see how two sprites will look when they overlap. Additionally, a page can be assigned to any sprite number, allowing you to change the priority of a sprite.

Once you're finished creating, sprites can be saved on disk. You can save individually or several sprites all at once. Options for save are as Byt statement for your assembler, or as PRG files – the option for saving as Basic DATA statements has not been implemented but wouldn't be hard to add.

Editor's Note

Some of the reverse field characters in Paul's program are obtained using the CTRL key. Specifically, the 'c' in line 5050, the 'i' in line 5100, and the 'd' in line 5130. Also, please be careful of ones (1) and Ls (l). Their typeset versions appear very similar and are easily confused. For comparison, you might refer occassionally to line 6540.

Use the Cursor keys to move around the palette.
1 erase pixel under cursor
2 use colour #1 to turn pixel under cursor on
3 use colour #2 to turn pixel under cursor on
4 use colour #3 to turn pixel under cursor on

: move current sprite displayed left
; move current sprite displayed right
@ move current sprite displayed up
/ move current sprite displayed down

X expand/contract current sprite horiz.
Y expand/contract current sprite vert.
↑I (CTRL–I) move sprite on palette down a row
↑D (CTRL–D) move sprite on palette up a row

CLR clears current palette
F1 change background colour
F3 change alternate colour #1
F5 change alternate colour #2
F7 change alternate colour #3

N next sprite (0 to 7)
P previous sprite
+ incr. page # of current sprite (192 to 255)
– decr. page # of current sprite

L load a sprite file from disk
S save sprites in disk file
from which page?
to which page?
<D>ata, <S>rc, <P>rg
where D = Basic Data statements
(not implemented)
S = assembler Byt statements
P = a PRG file
Filename?

C copy sprite definition on current page to which page? The target page # becomes the current page #.
↑C (CTRL–C) copy sprite definition on current page to next page. Also increments current page #.

```
100 rem sprite palette - multi-colour sprite editor
110 rem by paul higginbottom
120 rem
130 if lf=1 goto 6550
1000 v=13*4096 : cr=13*4096+8*256
1005 c0=0 : c1=1 : c2=2 : c3=3 : c4=4 : c5=5 : c6=6 : c7=7 : c8=8 : c9=9
1010 sc=1024 : sd=sc+1016 : pm=64 : ll=40 : sp=0 : md=0
1020 poke v+32,0 : poke v+33,0 : poke v+17,27+64 : poke v+37,2 : poke v+38,5q
1030 for i=0 to 7 : poke sd+i,192+i : poke v+39+i,c8+i
1060 a(i)=c2↑i : b(i)=255-a(i)
1070 next
1080 poke v+21,c0 : poke v+28,255
1090 for i=0 to 3 : c(i)=a(i*c2)+a(i*c2+c1)
1100 d(i)=255-c(i) : e(i)=a(i*c2) : next
1210 mx=11 : nx=12 : my=20 : ny=21
1220 poke 650,128 : gosub 8100
1300 for i=0 to 3 : q=i*c2 : x(q)=243 : y(q)=70+i*30
1305 x(q+c1)=270 : y(q+c1)=70+i*30 : next
1310 for sp=0 to 7 : gosub 7700 : next
1320 open 1,8,15
1400 ip=20 : rem input position
1410 ib$=" [space19] " :rem input blanking string
1900 poke v+21,1 : sp=0
2000 print " [Cyn] S ";
2010 for i=0 to 20 : print spc(24)" * " : next
2020 print " ********************** "
2030 print " q Colours:[space2]:[space2]:[space2]: Q "
2090 for i=1 to 3 : q=sc+925+i*c3 : r=32+i*64 : poke q,r : poke q+c1,r : next
```

```
2100 gosub 9000
2120 pg = peek(sd + sp)
2130 p = pg*pm
2140 poke 247,sp
2150 sys mc
2600 print " s[Down22]sprite[space4][Lft4] " sp " [Lft], page[space4][Lft4] " pg
3000 r = sc + y*ll + x*c2 : t = peek(r)
3010 s = peek(r)
3020 s = (s + 64)and 255
3030 poke r,s : poke r + 1,s
3040 for i = 1 to 25 : get a$ : if a$ = "" then next : goto 3020
3050 poke r,t : poke r + 1,t
3090 if a$<> " ] " goto 3110
3100 x = x + c1 + nx*(x = mx) : if x = c0 goto 3125
3105 goto 3000
3110 if a$ = " [Left] " then x = x-c1-nx*(x = c0) : goto 3000
3120 if a$<> " q " goto 3130
3125 y = y + c1 + ny*(y = my) : goto 3000
3130 if a$ = " Q " then y = y-c1-ny*(y = c0) : goto 3000
3140 if a$< " 1 " or a$> " 4 " goto 3300
3150 c = val(a$)-c1
3160 r = p + y*c3 + x/c4 : q = peek(r) : bp = c3-(x and c3) : q = (q and d(bp)) or c*e(bp)
3165 poke r,q
3170 r = sc + y*ll + x*c2 : c = c*64 + 32
3180 poke r,c : poke r + c1,c
3190 goto 3100
3300 if a$<> " x " goto 3320
3310 q = a(sp) : w = peek(v + 29) : w = (w and not q) or (not w and q)
3315 poke v + 29,w : goto3000
3320 if a$<> " y " goto 3340
3330 q = a(sp) : w = peek(v + 23) : w = (w and not q) or (not w and q)
3335 poke v + 23,w : goto3000
3340 if a$<> " + " goto 3370
3350 pg = pg + c1
3360 poke sd + sp,pg : goto 2100
3370 if a$ = " - " and pg>192 then pg = pg-c1 : goto 3360
3380 if a$<> " n " goto 3390
3385 sp = (sp + c1) and c7 :goto 3400
3390 if a$<> " p " goto 5000
3395 sp = (sp-c1) and c7
3400 poke v + 21,peek(v + 21) or a(sp) : goto 2100
5000 if a$ = " E " then poke v + 33,(peek(v + 33) + 1)and 15 : goto 7000
5010 if a$ = " F " then poke v + 37,(peek(v + 37) + 1)and 15 : goto 7000
5020 if a$ = " G " then poke v + 39 + sp,(peek(v + 39 + sp) + 1)and 15 : goto 7000
5030 if a$ = " H " then poke v + 38,(peek(v + 38) + 1)and 15 : goto 7000
5040 if a$ = " S " then for i = 0 to 63 : poke p + i,0 : next : goto 2100
5050 if a$ = " c " then for i = 0 to 63 : poke p + 64 + i,peek(p + i) : next : goto 3350
5060 if a$<> " c " goto 5100
5070 p$ = " to which page? " : gosub 8200 : if i$ = "" goto 3000
5080 if q = 0 goto 7910
5090 for i = 0 to 63 : poke q*64 + i,peek(p + i) : next : pg = q
5095 p$ = " now at new page " : gosub 8400 : goto 3360
5100 if a$<> " i " goto 5130
```

```
5110 for i=p+60 to p step -1 : poke i+c3,peek(i) : next
5120 for i=0 to 2 : poke p+i,0 : next : goto 2100
5130 if a$<>"d" goto 5160
5140 for i=p+3 to p+63 : poke i-c3,peek(i) : next
5150 for i=61 to 63 : poke p+i,0 : next : goto 2100
5160 if a$=";" then x(sp)=(x(sp)+c1) and 511 : gosub 7700 : goto 3000
5170 if a$=":" then x(sp)=(x(sp)-c1) and 511 : gosub 7700 : goto 3000
5180 if a$="@" then y(sp)=(y(sp)-c1) and 255 : gosub 7700 : goto 3000
5190 if a$="/" then y(sp)=(y(sp)+c1) and 255 : gosub 7700 : goto 3000
6000 if a$<>"s" goto 6500
6010 p$="from which page?" : gosub 8200 : if i$="" goto 3000
6020 fp=q : if q=0 goto 7910
6030 p$="to which page?" : gosub 8200 : if i$="" goto 3000
6040 lp=q : if q=0 goto 7910
6045 if lp<fp goto 7940
6050 p$="<d>ata,<s>rc,<p>rg?" : gosub 8200 : if i$="" goto 3000
6060 if (i$<>"s") and (i$<>"p") goto 7920
6065 if i$="s" goto 6200
6070 gosub 8500 : if i$="" goto 3000
6080 open 2,8,1,i$ : gosub 7800 : if p$<>"ok" goto 7990
6085 p$="saving - wait" : gosub 8400
6090 q=fp*pm : print#2,chr$(q-int(q/256)*256)chr$(q/256);
6100 for i=q to lp*pm+63 : print#2,chr$(peek(i)); : next
6110 close2 : p$="done" : goto 7990
6200 gosub 8500 : if i$="" goto 3000
6210 open 2,8,2,i$+",s,w" : gosub 7800 : if p$<>"ok" goto 7990
6220 p$="writing source..." : gosub 8400
6230 q=fp*pm : c=lp*pm+63
6240 print#2,";sprite data" : print#2,";"
6250 for i=q to c step 8 : print#2,".byt ";
6260 for j=0 to 7:print#2,mid$(str$(peek(i+j)),c2);:if j<>c7 then print#2,",";
6270 next : print#2 : next : print#2,";" : print#2,".end" : goto 6110
6500 if a$<>"l" goto 6600
6510 gosub 8500 : p$="loading - wait" : gosub 8400
6540 lf=1 : load i$,8,1
6550 goto 2000
6600 if a$="o" then poke v+21,peek(v+21) and b(sp) : goto 3385
6900 goto 3000
7000 gosub 9000 : goto 3000
7700 rem set sprite position
7710 poke v+sp*c2,x(sp) and 255 : q=peek(v+16) and b(sp)
7720 if x(sp)>255 then q=q or a(sp)
7730 poke v+16,q : poke v+sp*c2+c1,y(sp)
7760 return
7800 input#1,a$,p$,a$,a$ : return
7900 rem error out
7910 p$="less than 192" : goto 7990
7920 p$="not implemented" : goto 7990
7930 p$="say what?" : goto 7990
7940 p$="to<from!" : goto 7990
7990 gosub 8400 : goto 3000
```

```
8000 data 169,   0, 133, 248, 133, 251, 169,   4
8005 data 133, 249, 166, 247, 189, 248,   7, 133
8010 data 250, 160,   6,   6, 250,  38, 251, 136
8015 data 208, 249, 169,  21, 133, 254, 169,   0
8020 data 133, 253, 164, 253, 192,   3, 208,  27
8025 data  24, 165, 248, 105,  16, 133, 248, 144
8030 data   2, 230, 249,  24, 165, 250, 105,   3
8035 data 133, 250, 144,   2, 230, 251, 198, 254
8040 data 208, 220,  96, 177, 250, 133, 247, 169
8045 data   3, 133, 252, 162,   0,   6, 247, 144
8050 data   2, 232, 232,   6, 247, 144,   1, 232
8055 data 169,  32, 224,   0, 240,   6,  24, 105
8060 data  64, 202, 208, 250, 160,   0, 145, 248
8065 data 200, 145, 248,  24, 165, 248, 105,   2
8070 data 133, 248, 144,   2, 230, 249, 198, 252
8075 data  16, 209, 230, 253, 208, 164,  -1
8100 mc=4*4096 : m=mc : ne=-1
8102 read a,b : if peek(mc)=a and peek(mc+c1)=b then return
8105 restore
8110 read a : if a<>ne then poke m,a : m=m+c1 : goto 8110
8120 return
8200 rem input routine
8210 gosub 8400 : print spc(ip):i$=""
8220 f=0
8230 print "*[Lft]"; : goto 8250
8240 print " [Lft]";
8250 for i=1 to 30:get a$ : if a$="" then next:f=c1-f : on f+c1 goto 8230,8240
8260 if a$=chr$(20) and i$="" goto 8250
8270 if a$=chr$(20) then i$=left$(i$,len(i$)-c1) : print a$; : goto 8220
8280 if a$<>chr$(13) goto 8320
8290 print "[Down22]" spc(ip)ib$
8300 print spc(ip)ib$:q=0:if val(i$)>191 then q=val(i$)
8310 return
8320 if a$<chr$(32) or a$>"z" goto 8220
8330 if len(i$)<10 then i$=i$+a$ : print a$;
8340 goto 8220
8400 rem prompt
8410 print "[Down22]" spc(ip)ib$ : print "[Q]" spc(ip)p$ : return
8500 rem input filename
8510 p$="filename?" : gosub 8200 : return
8999 end
9000 rem copy sprite colours into extended colour mode registers
9010 poke v+34,peek(v+37)
9020 poke v+35,peek(v+39+sp)
9030 poke v+36,peek(v+38)
9040 return
```

Commodore 64
Graphics Utility

Brad Templeton
Waterloo, Ont.

Writing programs for the Commodore 64 can become rather cumbersome when you want to include high resolution graphics. The Basic has no commands that allow direct processing of the screen, and although there are cartridges and programs that offer special graphics commands, there are still two problems; one, you have to pay for them; and two, anyone else who wants to run your programs must also have the same extended commands installed. This graphics utility gives you 5 of the more common functions for manipulating the colour graphics screen. It's short enough (about 750 bytes) that it can be included as part of any Basic program. And best of all, it's free!

The routines work in multi-colour mode (sometimes called "medium-res") rather than actual high-res or "bit map" mode. Bit map mode offers better resolution, but Multi-colour mode allows for more colours – 4, as opposed to 2 in bit-map. Besides, the resolution is still pretty sharp – most games are programmed in multi-colour.

The plotting resolution is 160 horizontal (0–159) by 200 vertical (0–199). Combining these plotting commands with Basic programs to generate co-ordinates for, say, trigonometric functions makes the display task the easiest part. Adding POKEs to alter other VIC II chip registers will give some spectacular results. With sprites, collision detect, and the other features, the combinations are limitless.

About The Program

The first 15 bytes are jump vectors to the 5 commands. It's been assembled to reside in the 4K of RAM that sits between the Basic Interpreter ROM and the I/O chip. If you have other machine language that normally sits there, the source code (PAL format) is available from the Toronto PET Users Club so you can re-assemble it – it is not relocatable as is.

The vectors start at 49153. Setting a variable like BA (Base Address) to 49153 means you can easily access each command with a SYS BA plus some multiple of three between 0 and 12 (ie. SYS BA, SYS BA+3, +6, etc.), or you could use other variable to store the addresses of each vector.

Parameters for each command can be explicit numbers, variables, or any expression that yields numeric results. RND, LEN, ASC, SIN, ATN are just a few. Use your imagination.

The Commands

Initialize – SYS 49153 or SYS BA

Format: sys ba, c0, c1, c2, c3

Sets up the colour graphics screen with 4 colours:

c0 : background	(53281)
c1 : alternate background #1	(53282)
c2 : alternate background #2	(53283)
c3 : goes into the colour nybbles	

When SYS BA is called, the screen is cleared to the background colour. In subsequent commands, the 4 colours chosen here can now be referenced using the numbers 0 to 3. Border colour must be set with POKE 53280, C.

Read Location – SYS 49156 or SYS BA+3

Format: sys ba+3, row, column

Reads the given pixel. The colour, from 0 to 3, in that location will be stored in location 256 for you to PEEK at.

Set Point – SYS 49159 or SYS BA+6

Format: sys ba+6, row, column, type

Turns the given pixel on or off. The type is a number from 0 to 3 and shows one of the four colours chosen in the initialize routine. Choose type 0 to turn point off (ie. background colour).

Restore Text Mode – SYS 49162 or SYS BA+9

Format: sys ba+9 (: print " S ")

Gives you back the normal text screen. The screen will be full of garbage from the colour graphics screen, so it's best to follow with a print clear screen.

Draw Line – SYS 49165 or SYS BA+12

Format: sys ba+12, row1, column1, row2, column2, type

Draws a line from the point at row1/column1 to the point at row2/column2. The type, 0 to 3, sets the colour of the line. Again 0 = off. Example:

sys ba+12, 0, 0, 199, 159, 3

. . .draws a diagonal line from the top left corner to the bottom right in colour 3. Use the same command replacing ",3" with ",0" to erase the line.

Setting Up

The sample program draws a 3–D box. Line 90 calls the subroutine at line 1000 to POKE the program into memory. This is how you would include it in transportable programs, but for your own use you may wish to write a program file, if you have disk, and LOAD it from your program. To make a binary file of the utility, enter lines 1000 onward first. Then make these changes/additions.

Add: 900 open 8, 8, 8, "@0:g routines,p,w"
Chg: 1000 . . . 49853 changes to 49855
Chg: 1010 . . . poke j, x to print#8, chr$(x);
Add: 1025 close 8
Chg: 1030 . . . ch<>84065 to ch<>84257
Add: 1045 data 0, 192

With the file in place, line 90 of the demo program would be replaced by:

```
90 if peek(49152)<>96 then load "g routines",8,1
```

The utility will only be loaded if the byte at 49152 is not 96. Subsequent RUNs will therefore not attempt to load it over again. Don't forget to include ",1" at the end to specify a direct load file.

Sample Program

```
 90 gosub 1000
100 ba=49153 : k=48 : dr=ba+12
110 sys ba, 9, 3, 0, 1
120 for x=0 to 48 step 8 : p=3+(x=48)
130 sys dr, x, x, x, 160-x, p
140 sys dr, x, x, 192-x, x, p
150 sys dr, x, 160-x, 192-x, 160-x, p
160 sys dr, 192-x, x, 192-x, 160-x, p
170 next x
180 sys dr, 0, 159, 192, 159, 3
190 sys dr, 193, 0, 200, 0, 0
200 for x=0 to 8
210 sys dr, 0, 20*x, k, k+8*x, 3
220 sys dr, 146, k+8*x, 192, 20*x, 3
230 sys dr, k, k+8*x, 146, k+8*x, 2
240 next x
250 for x=1 to 11
260 sys dr, x*16, 0, 48+x*8, 48, 3
270 sys dr, k+8*x, k, k+8*x, 112, 2
280 sys dr, k+8*x, 112, x*16, 160, 3
290 next x
300 get i$ : if i$="" then 300
310 sys ba+9 : print " S "
320 end
1000 for j=49152 to 49853
1010 read x : ch=ch+x : poke j, x
1020 next
1030 if ch<>84065 then print "data error" : stop
1040 return
1050 data  96,  76,  76, 193,  76, 169
1060 data 193,  76, 143, 193,  76, 192
1070 data 193,  76, 217, 193,  76,  90
1080 data 192,  76, 102, 192,  76,  24
1090 data 193, 169,   0, 133,  25, 169
1100 data  32, 133,  26, 169,   0, 168
1110 data 162,  32, 145,  25, 200, 208
1120 data 251, 230,  26, 202, 208, 246
1130 data 169,   0, 133,  25, 133,  27
1140 data 169,   7, 133,  26, 160, 231
1150 data 169, 219, 133,  28, 162,   4
1160 data 165, 251, 145,  27, 165, 252
1170 data 145,  25, 136, 192, 255, 208
1180 data 243, 198,  26, 198,  28, 202
1190 data 208, 236,  96, 169, 255,  96
1200 data 165, 251, 201, 200, 176, 247
```

```
1210 data 165, 252, 201, 160, 176, 241
1220 data 152,  72,  32, 136, 192, 160
1230 data   0, 165, 252,  74,   8,  74
1240 data 177,  25, 176,   4,  74,  74
1250 data  74,  74,  40, 176,   2,  74
1260 data  74,  41,   3, 133, 163, 104
1270 data 168, 165, 163,  96, 165, 251
1280 data  74,  74,  41, 254, 168, 185
1290 data 180, 192, 133,  25, 185, 181
1300 data 192, 133,  26, 165, 251,  41
1310 data   7, 133, 163, 165, 252,  41
1320 data 252,  10, 144,   2, 230,  26
1330 data  24,   5, 163, 101,  25, 133
1340 data  25, 144,   2, 230,  26,  96
1350 data   0,  32,  64,  33, 128,  34
1360 data 192,  35,   0,  37,  64,  38
1370 data 128,  39, 192,  40,   0,  42
1380 data  64,  43, 128,  44, 192,  45
1390 data   0,  47,  64,  48, 128,  49
1400 data 192,  50,   0,  52,  64,  53
1410 data 128,  54, 192,  55,   0,  57
1420 data  64,  58, 128,  59, 192,  60
1430 data   0,  62,   0,   4,  40,   4
1440 data  80,   4, 120,   4, 160,   4
1450 data 200,   4, 240,   4,  24,   5
1460 data  64,   5, 104,   5, 144,   5
1470 data 184,   5, 224,   5,   8,   6
1480 data  48,   6,  88,   6, 128,   6
1490 data 168,   6, 208,   6, 248,   6
1500 data  32,   7,  72,   7, 112,   7
1510 data 152,   7, 192,   7, 133, 164
1520 data 152,  72,  32, 136, 192, 165
1530 data 252,  41,   3, 168, 185,  72
1540 data 193,  72, 165, 252,  74,   8
1550 data  74, 165, 164, 176,   4,  10
1560 data  10,  10,  10,  40, 176,   2
1570 data  10,  10, 133, 164, 104, 160
1580 data   0,  49,  25,   5, 164, 145
1590 data  25,  76, 129, 192,  63, 207
1600 data 243, 252,  32, 253, 174,  32
1610 data 158, 183, 142,  33, 208,  32
1620 data 253, 174,  32, 158, 183, 138
1630 data  10,  10,  10,  10, 133, 164
1640 data  32, 253, 174,  32, 158, 183
1650 data 138,  41,  15,   5, 164, 133
1660 data 252,  32, 253, 174,  32, 158
1670 data 183, 134, 251, 173,  17, 208
1680 data   9,  32, 141,  17, 208, 173
1690 data  22, 208,   9,  16, 141,  22
1700 data 208, 169,  28, 141,  24, 208
1710 data  76,  25, 192,  32, 253, 174
1720 data  32, 158, 183, 134, 251,  32
1730 data 253, 174,  32, 158, 183, 134
1740 data 252,  32, 253, 174,  32, 158
1750 data 183, 138,  76,  24, 193,  32
1760 data 253, 174,  32, 158, 183, 134
1770 data 251,  32, 253, 174,  32, 158
1780 data 183, 134, 252,  32,  90, 192
1790 data 141,   0,   1,  96, 173,  17
1800 data 208,  41, 223, 141,  17, 208
1810 data 173,  22, 208,  41, 239, 141
1820 data  22, 208, 169,  20, 141,  24
1830 data 208, 169,   0,  24,  96, 169
1840 data   0, 133, 171, 133, 174,  32
1850 data 253, 174,  32, 158, 183, 134
1860 data 166,  32, 253, 174,  32, 158
1870 data 183, 134, 168,  32, 253, 174
1880 data  32, 158, 183, 138,  56, 229
1890 data 166, 176,   6,  73, 255, 105
1900 data   1, 198, 174, 133, 173,  32
1910 data 253, 174,  32, 158, 183, 138
1920 data  56, 229, 168, 176,   6,  73
1930 data 255, 105,   1, 198, 171, 133
1940 data 170,  32, 253, 174,  32, 158
1950 data 183, 134, 176, 165, 173, 197
1960 data 170, 176,   2, 165, 170, 133
1970 data 175, 169,   0, 133, 163, 165
1980 data 170, 133, 164,  32, 162, 194
1990 data 165, 163, 133, 169, 165, 164
2000 data 133, 170, 169,   0, 133, 163
2010 data 165, 173, 133, 164,  32, 162
2020 data 194, 165, 163, 133, 172, 165
2030 data 164, 133, 173, 169, 128, 133
2040 data 167, 133, 165, 162, 169,  32
2050 data 140, 194, 162, 172,  32, 140
2060 data 194, 166, 175, 165, 168, 133
2070 data 252, 165, 166, 133, 251, 165
2080 data 176,  32,  24, 193, 165, 167
2090 data  24, 101, 169, 133, 167, 165
2100 data 168, 101, 170, 133, 168, 165
2110 data 165,  24, 101, 172, 133, 165
2120 data 165, 166, 101, 173, 133, 166
2130 data 202, 208, 214,  96, 181,   2
2140 data  16,  17,  56, 181,   0,  73
2150 data 255, 105,   0, 149,   0, 181
2160 data   1,  73, 255, 105,   0, 149
2170 data   1,  96, 162,  16, 169,   0
2180 data   6, 164,  42, 176,  14, 197
2190 data 175, 144,   2, 229, 175,  38
2200 data 163,  38, 164, 202, 208, 240
2210 data  96, 229, 175,  56, 176, 243
```

Raster Interrupts On The Commodore 64 Part 1: Introduction and Theory

Dave Berezowski
Toronto, Ont.

One feature of the Commodore–64 that is probably least understood and most powerful is the ablility to "muck–about" with the raster beam. Many things are possible such as, continuous smooth scrolling of the entire screen, mixed–graphic modes (ie. text and hi–res), flicker–free animation, more than 8 sprites on the screen at one time, etc. In this and subsequent articles, I will attempt to explain how raster interrupts work and how you can use them. I'll assume that you have first learned the basic concept of interrupt coding, and how the other features of the VIC II chip work, such as background colour, border colour, etc. Raster interrupts will allow you to use these features to obtain more interesting effects in your programs.

Let's stop here for a moment and explain some terms we'll need to know. The first thing that you need to know is how the picture gets drawn on your T.V. or monitor. Imagine that the screen is a grid of 320 dots across by 200 dots down. Each dot, sometimes called a "pixel", is made of a phosphorescent material that glows when struck by electrical energy. Colour pixels are made of three bits of phosphor; red, green, and blue. This combination allows for all the colours by hitting each dot with a controlled amount of energy. However, to avoid potentially confusing details concerning colour theory, we'll assume each dot will glow in the same colour, or simply "on" or "off".

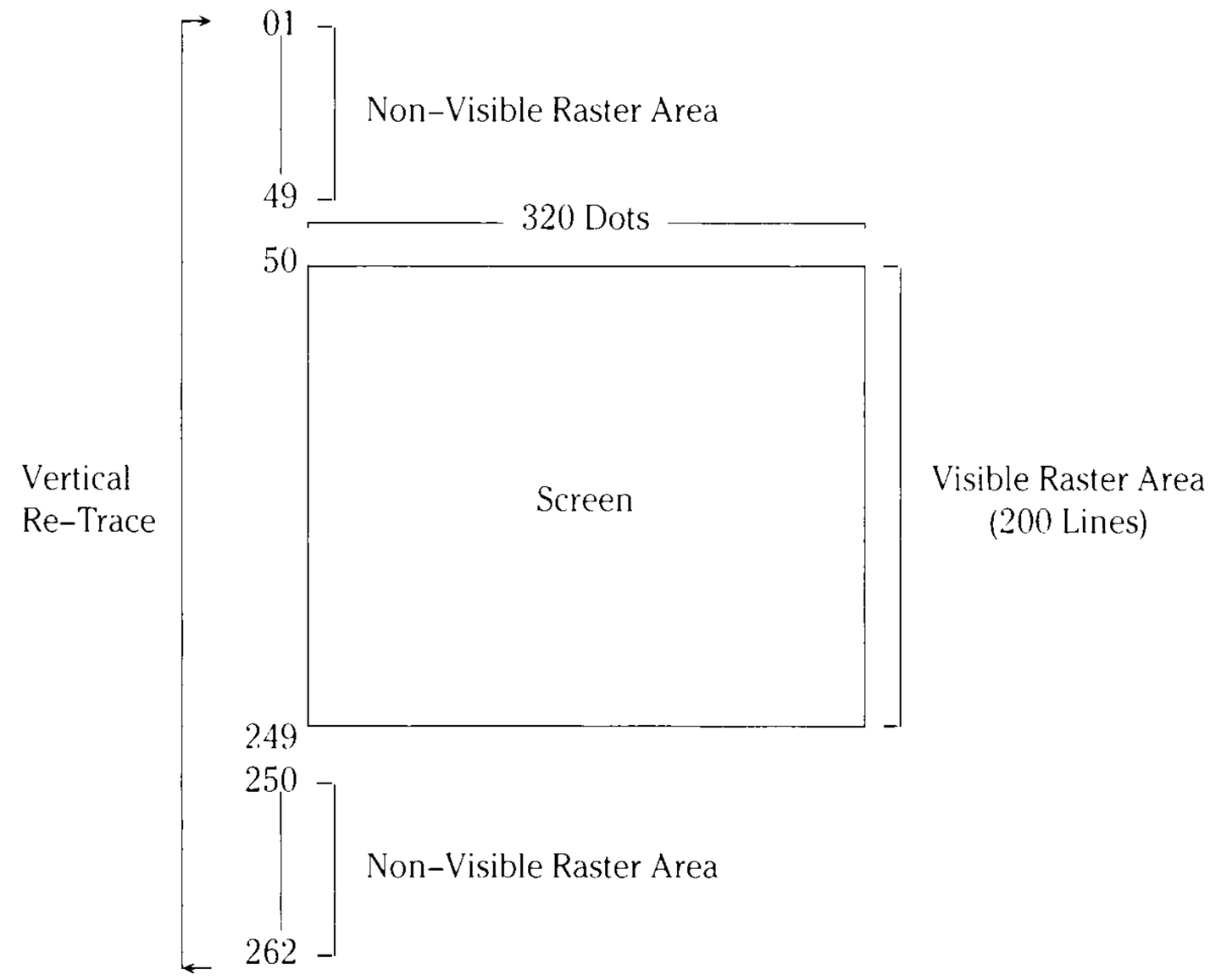
Energy is supplied to each dot by the cathode ray or electron beam that is controlled by that large cylindrical object on the back of the picture tube. Most of you have probably seen this; it's covered with several wrappings of copper wire and referred to as the electron gun. The electron gun can direct the beam at any dot on the screen. By rapidly switching the beam on or off, each dot either glows or does not glow. Since the Cathode Ray Tube (CRT) is not very smart, it doesn't know that it's drawing letters, numbers, sprites, etc. It only knows that it's turning dots on or off. The end result is a pattern of on and off dots that forms the picture that we see, something we all merely take for granted.

The beam starts on the top line (line 1) of the screen and "draws" 320 dots across from left to right. Then it zips back to the beginning of line 2, and draws 320 more dots. Each line of 320 dots is known as a "raster". The beam essentially draws a whole line in one swoop, so we refer to it as the "raster beam", a term that fits very nicely since we're dealing with raster interrupts.

A "raster interrupt", therefore, does exactly what the term implies. It allows us to "interrupt" the process of drawing the entire screen and perform some operation in–between two "rasters". The two can be ANY two of your choosing, and the operation, for example, might be a change of background colour, so that the top half of the screen has a background colour of red, and the bottom half is in blue. We'll be looking at this later, but taking it one step further, we could change the background colour back to red (or another colour) just before reaching the bottom of the screen. And ultimately, we could change the background colour for every raster on the screen. All this and more is possible using the raster interrupt feature.

Once again then, the raster beam starts drawing rasters from line 1 at the top of the screen, until it gets down to the bottom of the screen at line 200. After it finishes drawing line 200 it zips back up to the top of the screen (known as "vertical retrace") and starts all over again. It draws a complete picture on your screen 60 times per second. So, we could say that every line gets re–drawn once every 1/60th of a second or every 16,667 micro–seconds (assuming that the Commodore 64 has a clock speed of 1Mhz.) Lets take a pictorial look at whats happening.

Figure 1.



Now, I should explain something here so as not to confuse you. The **full–range** of the raster beam is from raster line 1 (RL1) to raster line 262 (RL262). However, the visible raster lines are from RL50 to RL249. What this means is that when the raster beam is drawing the first line of dots on your screen, it is really on its 50'th line. Therefore, raster lines 1–49 and 250–262 are actually drawn off screen and make up what I referred to earlier as the "vertical–retrace". (Incidentally, the raster beam values and the sprite positions are the same. So when a sprite is at y position 51, it is also on the 51'rst raster beam.)

The Raster Beam Registers

Now that you understand (I hope) how the raster beam works, I can now introduce some of the registers of the video chip that we'll need to use. Familarize yourself with register 17 ($D011 or 53265) and register 18 ($D012 or 53266) which are dual purpose registers. When you read them, they return the current position of the raster beam (be it on or off screen). When you write to them, it sets up a raster–compare value which can generate a raster interrupt when the raster beam position equals the number that you put in the register. It is like saying to the video chip, "Tell me when the raster beam is at position xxx".

Note that because the raster beam can have a value greater than 255, we need 9 bits to tell us its position. So, the 8th bit of register 17 ($D011 or 53265) is used and is actually the 9th bit of the raster value. Naturally, bits 1–8 of register 18 are bits 1–8 of the raster beam. To read the raster beam you could perform the following:

```
10 rv = peek(53266) + (peek(53265)and128)*256
20 print " raster is at line " ;rv
```

To write to the raster register to set up a raster compare value we could code: (remember, we must not disturb the other bits in register 17)

```
10 rc = 251 : rem end of screen, for example
20 poke 53266, rc and 255 : rem put lower 8 bits in reg 17
30 poke 53265, (peek(53265)and127) or (int(rc/256)*128)
40 print " raster compare now set at line " ; rc
```

The next two registers we have to be aware of are registers 25 ($D019 or 53273) and 26 ($D01A or 53274). Register 25 is the interrupt status register. It tells us who has generated an interrupt on the IRQ line (Interrupt ReQuest). The interrupts can come from the light–pen, sprite–sprite collisions, sprite–background collisions, or raster compare matches. In this

case we are only concerned with raster compare interrupts and need only look at bit 0 of register 25. Here is a picture and description of register 25.

Figure 2 : Register 25 ($D01A or 53274)
Interrupt Status/Clear Register

BIT 7 : – has anybody requested an IRQ?
BIT 3 : – has a light–pen IRQ been requested?
BIT 2 : – has a sprite–to–sprite IRQ been requested?
BIT 1 : – has a sprite–to–background IRQ been requested?
BIT 0 : – has a raster–compare IRQ been requested?

A "0" in the corresponding bit position means that an IRQ has not been requested, whereas a "1" means that an IRQ has been requested.

This register is also used to clear the interrupt status. You must write a "1" back to the corresponding bit of the interrupt that you serviced. Example: Let's say that a raster compare interrupt occurs and that you service (take care of) it and return. If you forget to tell the video chip that you serviced the interrupt, as soon as you exit from the interrupt, the CPU will think the video chip has another interrupt needing service. Thus your machine will just sit there and constantly interrupt itself. (From your point of view, the machine will have appeared to have crashed!) So as I said, to clear the interrupt you must write a 1 to the bit of the interrupt that you serviced. In this case we would code:

poke 53274, peek(53274) or 1

Register 26 is the interrupt enable register. It tells the system which interrupt it should recognize. That is, a raster compare interrupt could occur, BUT if you haven't enabled it, then it will be ignored. For convenience, register 26 is laid out the same as register 25 with respect to the ordering of the different types of interrupts. Refer to figure 3 for specifics.

Figure 3 : Register 26 ($D01B or 53275)
Interrupt Enable Register

BIT 3 : – are light–pen IRQs enabled?
BIT 2 : – are sprite–to–sprite IRQs enabled?
BIT 1 : – are sprite–to–background IRQs enabled?
BIT 0 : – are raster–compare IRQs enabled?

A "0" in the corresponding bit position means that an IRQ has not been enabled, whereas a "1" means that an IRQ has been enabled.

One more register to worry about and then we can get into an example. First some background info. Every 1/60 of a second the 6526 (CIA #1) located at $D000–DC0F (56320–56375) generates an interrupt on the IRQ line. It does this so that your C64 can flash the cursor, update the time–clock, check the stop–key, read the keyboard, etc. So unless we are working deep within machine language and don't need BASIC, we better make sure that this routine is still being called, or else we won't have a cursor, can't press the stop–key, etc. We are left with three choices when dealing with raster–interrupts in what to do with the normal interrupt as follows:

1) We can leave this routine alone and allow it to occur normally every 1/60 of a second. But now we are going to have two interrupts occuring every 1/60 of a second, our raster compare IRQ and the normal IRQ. There's nothing wrong with this, just so long as you check which interrupt occured so that you can service the proper one.

2) We can disable the normal IRQ, but jump to it after we have finished doing our stuff. So we will have only one interrupt occuring but it will be forming two functions. (We must jump to the normal IRQ routine every 1/60 of a second if we want the use of the cursor, stop key, etc.). This is the method I chose in the program that follows the article.

3) If we don't need BASIC at all. then we can disable the normal IRQ routine and instead of jumping to it, exit our routine with the following code:

PLA : TAY : PLA : TAX : PLA : RTI.
(note: this piece of coding is located at $FEBC)

Anyway, we better get back to the CIA. The part of the CIA #1 we need to worry about is $DC0E (56334). It controls all sorts of things, but the main thing is that bit 0 controls whether timer A is running or not. Timer A counts down to zero every 1/60 of a second and generates an interrupt. If we stop timer A, then no normal IRQ interrupts will occur. Refer to figure 4 for specifics.

Figure 4:
CIA #1 Control Register ($DC0E or 56334)

7 Time of day clock freq : 1–50Hz, 0–60Hz
6 Serial port I/O mode : 1–output, 0–input
5 Timer A counts : 1–cnt sgnls, 0–system 02 sgnls
4 Force load timer A : 1–yes, 0–no
3 Timer A run mode : 1–one, 0–continuous
2 Timer A output on PB6 : 1–toggle,0–pulse
1 Timer A output on PB6 : 1–yes, 0–no
0 Start/Stop Timer A : 1–start, 0–stop

To stop timer A we code:

```
poke 56334, peek(56334) and 254
```

To start timer A we code:

```
poke 56334, peek(56334) or 1
```

NOTE: I must point out that all of the examples I have been giving you in BASIC are for readability sake only. BASIC is far too slow for working with raster interrupts. You must code in machine language. For example:

```
poke 56334, peek(56334) and 254
```

would become. . .

```
lda $dc0e
and #254
sta $dc0e
```

Lets leave all the technical stuff for now and apply what we have learned. Below is a program (a BASIC loader and the documented assembly source file) to give us a two-colour screen. The top half color of the screen can be controlled by poke 24582,ch and the bottom half by poke 24581,cl (ch and cl range from 0-15). Refer to figure 5 to see whats going on.

Next Issue:

A) For you technical types, I'll go into the timing constraints of raster interrupts (time taken to draw one raster line, total cycle time, vertical-retrace time, etc.) and why you need to know this stuff.

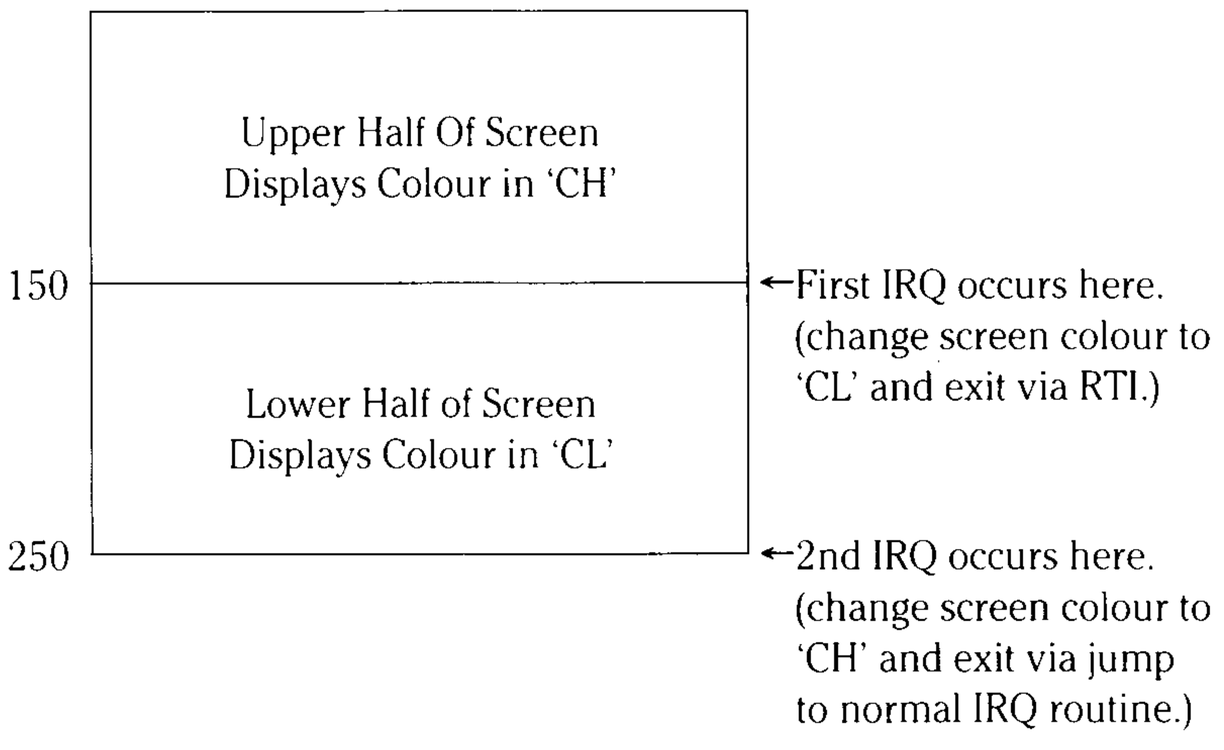
B) How to get a mixed-mode screen. We'll end-up with 20 lines of hi-res and 5 lines of text on the same screen.

C) ???

In future articles, I'd like to go into the art of smooth scrolling, flicker-free animation, more than 8 sprites on one screen, ???.

Digest what you learned for now and I'll see you next issue.

Figure 5.



```
100 rem raster.bas
110 gosub 1000
120 z=6*4096 : sys z
130 r1=z+4 : r2=z+3 : r3=z+6 : r4=z+5
140 c1=150 : c2=250 : c3=0 : c4=1
150 get a$ : if a$="" then 150
160 if a$="[Q]" then c1=c1+1 : goto 230
170 if a$="[Q]" then c1=c1-1 : goto 230
180 if a$="[ ]" then c2=c2+1 : goto 230
190 if a$="[Lft]" then c2=c2-1 : goto 230
200 if a$="[E]" then c3=c3+1and15 : goto 230
210 if a$="[F]" then c4=c4+1and15 : goto 230
220 goto 150
230 poke r1,c1 : poke r2,c2 : poke r3,c3 : poke r4,c4 : goto 150
240 :
250 rem cursor-up moves r1 up
260 rem cursor-down moves r1 down
270 rem cursor-right moves r2 down
280 rem cursor-left moves r2 up
290 rem f1 bumps top color
300 rem f3 bumps bottom color
1000 for j=24576 to 24698
1010 read x : ch=ch+x
1020 poke j, x : next
1030 if ch<>13196 then print "data error" : stop
1040 return
1050 data  76,   9,  96, 170, 170, 170, 170, 170
1060 data 170, 120, 169,   0, 141,  14, 220, 173
1070 data  20,   3, 141,   7,  96, 173,  21,   3
1080 data 141,   8,  96, 169,  78, 141,  20,   3
1090 data 169,  96, 141,  21,   3, 169, 150, 141
1100 data   4,  96, 169, 250, 141,   3,  96, 173
1110 data  17, 208,  41, 127, 141,  17, 208, 173
1120 data   4,  96, 141,  18, 208, 169,   0, 141
1130 data   6,  96, 169,   1, 141,   5,  96, 169
1140 data   1, 141,  26, 208,  88,  96, 172,  18
1150 data 208, 162,   2, 204,   3,  96, 240,   7
1160 data 136, 204,   3,  96, 240,   1, 202, 189
1170 data   4,  96, 141,  33, 208, 189,   2,  96
1180 data 141,  18, 208, 173,  25, 208, 141,  25
1190 data 208, 224,   1, 208,   3,  76, 188, 254
1200 data 108,   7,  96
```

```
;*****************************************************
;* routine to split the display screen to create a two-color      *
;* background. rast1 is the raster-compare value for the first    *
;* interrupt and determines where the bottom screen color will    *
;* start, the color is determined by botcol. rast2 is the         *
;* raster-compare value for the second interrupt and switches the *
;* background color to topcol at that point. if rast2 is less then*
;* 250 then the rest of the bottom of the screen as well as the top*
;* of the screen (up to rast1) will have a color of topcol.       *
;*****************************************************
;
;constants
;
irqvec    = $0314          ;irq jump vector.
vicii     = $d000          ;start of vicii chip.
rasthi    = vicii+17       ;raster compare hi.
rastlo    = vicii+18       ;raster compare low.
vicirq    = vicii+25       ;interrupt active.
vicena    = vicii+26       ;interrupt enable.
bgcol0    = vicii+33       ;background color.
ciacra    = $dc0e          ;cia control register a.
rtiext    = $febc          ;rti exit routine.
;
;
;initialize interrupt system (jump vector).
;
  *       = $6000
;
          jmp  init
;
;variables
;
rast2       * = *+1        ;second raster compare value (@ 24579).
rast1       * = *+1        ;first raster compare value (@ 24580).
;
botcol      * = *+1        ;bottom background color (@ 24581).
topcol      * = *+1        ;top background color (@ 24582).
;
irqsav      * = *+2        ;original irq vector.
;
init      sei              ;disable interrupts.
;
          lda  #0
          sta  ciacra      ;disable 6526 interrupts.
;
          lda  irqvec
          sta  irqsav
          lda  irqvec+1
          sta  irqsav+1    ;save original irq vector.
;
          lda  #<servic
          sta  irqvec
          lda  #>servic
```

```
        sta   irqvec+1    ;init irq vector to point to raster service code.
;
        lda   #150
        sta   rast1       ;init first raster-compare point.
        lda   #250
        sta   rast2       ;init second raster-compare point.
;
        lda   rasthi
        and   #%01111111
        sta   rasthi
        lda   rast1
        sta   rastlo      ;init raster compare to first raster value.
;
        lda   #0
        sta   topcol      ;init top color to black.
        lda   #1
        sta   botcol      ;init bottom color to white.
;
        lda   #1
        sta   vicena      ;enable raster compare interrupts.
;
        cli               ;enable interrupts.
        rts               ;return.
;
;main interrupt service code.
;
servic  ldy   rastlo
        ldx   #2          ;assume second interrupt.
        cpy   rast2       ;is this the second interrupt?
        beq   skip01      ;yes
        dey
        cpy   rast2       ;fudge for raster-beam timing error.
        beq   skip01      ;see explanation next article.
        dex               ;no, so set index for 1st interrupt.
;
skip01  lda   botcol-1,x
        sta   bgcol0      ;set background color.
;
        lda   rast2-1,x
        sta   rastlo      ;set raster-compare for next raster interrupt.
;
        lda   vicirq
        sta   vicirq      ;acknowledge that the interrupt was serviced.
;
        cpx   #1          ;is this the first interrupt?
        bne   exit2       ;no
;
exit1   jmp   rtiext      ;return from first interrupt.
;
exit2   jmp   (irqsav)    ;perform normal irq routine.
;
.end
```

The P128: A First Look, A Last Look

Fred Hambrecht
Chattanooga, TN.

Christmas in March! That's the way I felt unpacking my new Commodore B128 computer. A quick check of the reference manual. . . Whoops! No manual. Oh well, it's more fun without the book, besides I know Commodore inside and out, what could be different? Plug it in, turn it on, and the monitor (yes, monitor – no RF modulator in this one) comes up with the Commodore Basic message. They even redid the colour combination, white background with blue characters.

Let's push a few keys and see what happens. Push F1 on the top left and it printed "PRINT#". Each of the other keys has such goodies as "DIRECTORY", "DLOAD", "LIST", etc. Hm! Have to spend more time looking at that later. The keyboard has a really nice layout, with a typewriter style return key in the main array. The numeric pad contains a seperate enter key, along with a "?" key. In addition, a CE key is provided to clear entries to the last arithmetic operator entered or to the beginning of the line. A single as well as a double ZERO key should make any "bean counter" happy. The feel of the keyboard, in my opinion, is the best on the market today. Photograph number one shows the general layout.

The back of the unit has a few more surprises: two joystick ports, IEEE–488 interface, User Port, and a real honest–to–goodness RS–232 connector.

Enough of the outside, let's look to see if the Basic is the same. We entered a program in Basic to look at parse table in ROM and found a few more surprises. First of all, the Basic appears to be standard 4.0 with the following additions:

BANK — Sets the bank number (16 banks of 64K) for peeks, pokes, and bank save/load commands.

BLOAD/BSAVE — Used to load and save binary files from memory. I can't be sure but it appears that you can specify high and low addresses within the bank.

DISPOSE — Used with the TRAP command to purge unwanted NEXT and RETURNS.

IF..THEN..ELSE — A new one for Commodore.

INSTR — I have not been able to figure this one out.

PRINT USING — Both PRINT and PRINT# are supported.

PUDEF — This redefines the symbols used in PRINT USING

RESUME — Used with TRAP to RESUME after error handling.

TRAP — Uses user supplied error routines in place of the Commodore routines.

New Variables:

ERR$ — Reserved string for error messages.

EL — Reserved variable for error line.

ER — Reserved for error number.

TI$ — Returns time in the format HHMMSSJ where J is a 1/10th of a second. TI no longer exists!

KEY — Allows programming of the 20 function keys and takes the form KEY n, "expression" where n is the key number and "expression" is the string assigned to that key. KEY by itself will display a list of the current function key definitions.

The screen now resides (in bank 15) at locations 53248 to 54247 followed by the colour memory at locations 54272 to 55271. The screen has many of the features of the 8032. I have discovered Escape sequences that set windowing, insert line, delete line, erase to end/beginning of line, etc. Now I wish I had the manual.

Next, spend two hours on the phone calling the folks at Commodore. No one knows about the P128! Inside sources said the unit had been engineered "overseas" and they were as much in the dark as I was. Manuals are supposed to be available in the next month or so, and this unit was a dealer demo only. The unit was not supposed to be sold prior to FCC and UL approval.

I can't stand it any longer, get out the screwdriver and tear it apart. When the cover was removed, it revealed prototype ROMs. As can be seen in the photo, 28 pin ROMs have been *jerry rigged* on to 24 pin sockets. Several nice features. . . the power supply is a switching regulator type made by someone other than Commodore and is easily removed for service. The internal 128K of ROM is located in the lower left and upper right. Note the spaces for socketing an additional 128K already exist. Wonder if they plan to drill holes in the boards ala the old 16K PET? (Shh)

The processor is a 6509, functionally the same as the 6510 used in the Commodore 64, but with an additional bank select register that allows for the 16 banks of 64K. The addressing indicates the operating system, SID chip, I/O, etc., all reside in bank $F (15) and the internal memory is in banks 0, 1, and 2. Only bank 0 is used in an unexpanded 128. A feature that appears to exist is a Co-Processor function. Interrupt processor granted and refused signals leads me to think that we will probably see a multi-user system in the future.

Photo 2 shows the internal layout of my machine, and in spite of the fact that this is a prototype, a lack of wire jumpers, etc., indicates the hardware is pretty solid, and the one remaining "cat to skin" is the firmware. The model I am playing with has no tape routines. In fact, a "LOAD" will result with "DEVICE NOT PRESENT ERROR". Speaking of error messages, a new one exists. If you overflow the subroutine stack, the massage is "OUT OF STACK" rather than the old ambiguous "OUT OF MEMORY".

(Note: to Jim Butterfield, I stayed awake nights to try, just once, to generate a memory map before seeing yours in Compute!)

The memory may be viewed as 16 pages (0–15) of 64K each. Page 15 or $F is fixed in hardware as the operating system and hardware layed out as in Figure 1.

A further breakdown of $FD800 to $FE000 will let you see that the only difference between the P and the B is probably jumpers or land cuts. I suspect that changing a P128 to a B256 is no harder than changing a Fat 40 to an 8032. The B addresses appear in Figure 2.

Figure 1.

$FFFFF	
	Kernal ROM
$FE000	
	Port 2 6525
$FDF00	
	Port 1 6525
$FDE00	
	ACIA 6551
$FDD00	
	CIA 6526
$FDC00	
	Co Processor Ext. Port
$FDB00	
	SID 6581
$FDA00	
	Disk Port (internal)
$FD900	
	VIC II 6567
$FD800	
	Colour Memory 1K
$FD400	
	Screen RAM 1K
$FD000	
	Character ROM 4K
$FC000	
	Basic ROM Hi
$FA000	
	Basic ROM Lo
$F8000	
	External ROM
$F4000	
	Cartridge Slot
$F2000	

Figure 2.

$FEFFF	
	6525 Keyboard
$FDF00	
	6525 (IEEE-488)
$FDE00	
	6551 (RS-232 Port)
$FDD00	
	6526 (IEEE & User Port)
$FDC00	
	Ext. Processor
$FDB00	
	SID 6581
$FDA00	
	Disk Units (internal)
$FD900	
	CRT Controller 6545
$FD800	
	Screen RAM 2K
$FD000	
	Unused
$FC000	

A few general comments: The P128 should be a great machine. It looks smart, especially next to the new low profile disk drives. I am presently running this one with an 8250LP drive and the 8300P printer and am really pleased with it. I have been able to load most of the programs I have in Basic, and am in the process of converting some machine language programs. The only negative comment I have is that it is slower than the 64 and previous generation PET/CBMs. I believe the reason is that all 20 function keys appear to be processed by CHRGET in the run mode. I will reserve judgement on this until I see a final ROM set. Aside from that, it's the neatest Commodore yet!

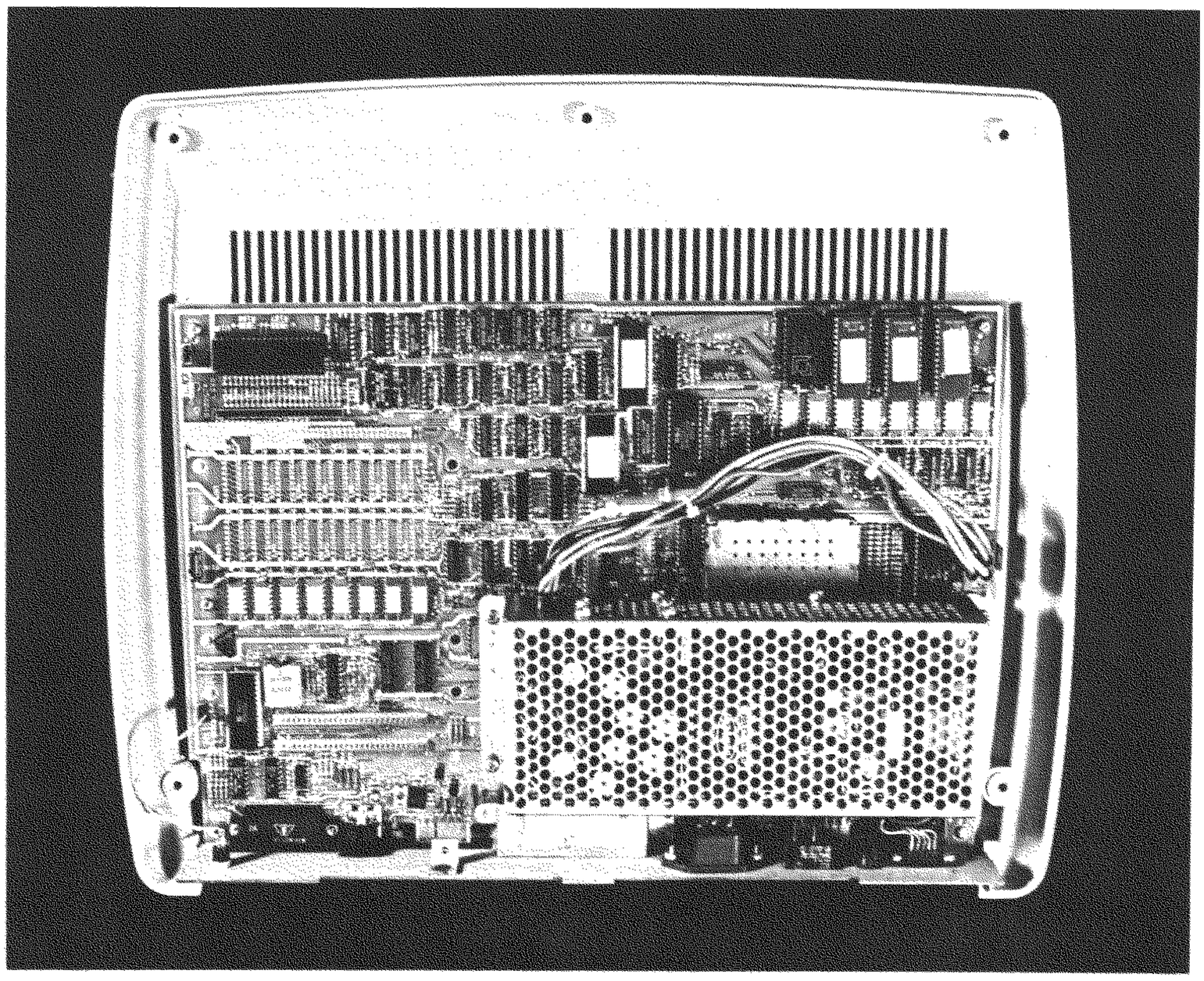
Editor's Note

First off, most of us, and Mr. Hambrecht, probably know by now that he has one of the few P128s ever released! This is the machine that was to have 40 column screen with colour, plus VIC II graphics and SID sound – like an *inflated* Commodore 64. But this project has been locked in the far depths of the Commodore dungeon. Mr. Hambrecht, you are now the proud owner, of what some may consider, a collectors' item!

I couldn't help publishing the article, however. After all, the B machine is identical to the P in all aspects except screen features. For example, you'll notice that the P has 1K set aside for screen RAM (40 cols by 25 lines) and another 1K for the corresponding colour RAM. The B has no colour, but 80 columns – therefore, 2K for screen RAM. The P uses the VIC II chip for video activity whereas the B uses the 6545 CRT Controller like in the 8032. This explains why the character ROM is addressable in the P and not in the B. Aside from these, the two machines differ very little.

You may be asking, "So why did Commodore cancel the P series? It seems like it would have been quite successful." Once again, I believe the best answer would come from the Commodore marketing mill. The market for the P is not as well defined as the rest of Commodore's line and with the price of the 64 dropping so low, the P becomes somewhat less attractive to the average consumer. Fear not though. The competition in the micro industry will see manufacturers continually releasing new machines. A successor to the 64 might not be in Commodore's immediate future, but you can bet it will happen.

Look for Jim Butterfield's B Series Memory Maps next issue!



B500 Machine Language: Transfer Sequences

Jim Butterfield
Toronto, Ont.

The Commodore B series can be programmed in Machine Language. But it takes a little more setup than has been required on previous Commodore computers. Memory is arranged in "banks" or "segments"; each segment may contain up to 64K of memory, and there may be up to sixteen segments. In the B256 series, banks 1 to 4 will contain 64K of RAM, and bank 15 (the "system" bank) contains ROM, I/O, and a little RAM. Other banks are not used.

This is a technical description of how the transfer sequences supplied by Commodore work, allowing a program to go from bank to bank as necessary. You don't need to know this in order to program in machine language on B series machines. This outline is for those who would like to look at the workings in detail.

Synchronized Code

Each bank of memory used for machine language must have code – a program – in similar locations. Bank 15 contains ROM, so the code is already in place. For the bank we choose to program – usually bank 3 – we must put the same code into identical addresses of RAM. The code is always the same; we usually load it from disk. Additionally, we'll need to load our machine language program in RAM. The "synchronized" code is vital to allow our program to flip between banks as needed.

Why would we want to go between banks? Well, suppose we wanted to print a character, for example. The routines to do this are in ROM, bank 15; our program is running in RAM, bank 3. We must move across to do the job.

Code must be synchronized for several reasons. First, when we make the actual bank jump we do it by change the bank number in address zero. We cannot change the address at which we are working; so the computer starts to execute in the new bank at the exact address it had reached in the old bank. Thus: old and new must work together. Secondly, we must pass information between banks, including subroutine addresses; this will be easier if all subroutines are located at the same place within their respective banks.

Starting Up

To start a machine language program, we must load the synchronized code into our selected bank. As previously noted, this will usually be bank 3. Next, we must load the machine language program we wish to run into bank 3. Finally, we need to get to this program. That's a whole story in itself. For the moment, we can summarize it by saying that BANK 3:SYS address will get you there.

Now: our program wishes to print something. As usual, it will load the character it wishes to print into the A register, and then will JSR $FFD2 to print. Our coding is the same. But what happens next is a whole new story.

For those who wish to track this technically, we'll give a specific example and work it through. At address hex 2000, we have the code to print the letter A: LDA #$41:JSR $FFD2. The JSR instruction is located at address 2002, and the moment we call it, we'll have a return address on the stack of 2004 (not 2005, oddly enough: the 6502 does JSR with a return offset of 1).

Our status so far: the stack contains a return address of 2004 and we find ourselves at address $FFD2.

The False Jump Table

Address FFD2 is part of what we recognize as the "Kernal Jump Table". Normally, we'd expect it to contain a Jump instruction to the specific routine to do the job, in this case, to print the character in the A register. But these are not normal circumstances; we're in bank 3 and the Kernal is in bank 15. What we find at address FFD2 is a subroutine jump: JSR $FEA7. In fact, the entire jump table in RAM contains exactly the same code: JSR $FEA7.

Stay with it. The return address – in this case, FFD4, goes to the stack. We now have two return addresses there. And now we're ready to tackle the code at $FEA7.

Let's follow the code from $FEA7

```
fea7  08       exsubf   php          ;save status
fea8  48                pha          ;save.a
fea9  a9 0f              lda  #15     ;system segment
feab  85 01              sta  i6509   ;set up ind pointer
fead  68                 pla          ;restore .a
feae  28                 plp          ;restore status
```

We've put our destination bank into the "indirect" bank control word, address 1. From this point on, any indirect indexed instruction will deal with data from bank 15.

```
feaf  4c b2 fe           jmp exsub
feb2  08       exsub     php          ;save status
feb3  78                 sei
feb4  48                 pha          ;.a
feb5  8a                 txa
feb6  48                 pha          ;.x
feb7  98                 tya
feb8  48                 pha          ;.y
feb9  20 19 ff           jsr  ipinit  ;init ipoint, stack
```

After a useless jump, we stack all our registers, lock out the interrupt, and call a subroutine which will set up an indirect address and load the "cross stack pointer" :

```
ff19  a0 01    ipinit    ldy  #1
ff1b  84 ad              sty  ipoint+1
ff1d  88                 dey
ff1e  84 ac              sty  ipoint      ;ipoint=$0100
ff20  88                 dey              ;.y =$ff
ff21  b1 ac              lda  (ipoint),y  ;load sp
ff23  60                 rts
```

Our indirect pointer, at hex AC and AD, is set to the stack page, hex 0100. Now we load the contents of 01FF from bank 15 (indirect address, remember?)

Address 01FF in bank 15 contains the stack pointer from the bank 15 stack. When we left bank 15 to come to our program, we left the pointer in that address. In fact, when we leave bank 3 soon, we'll leave the bank 3 stack pointer in 01FF here.

Now we can return from our subroutine and continue the main operation. Right now, the stack pointer for bank 15 is in the A register: we'll move it to Y.

```
febc  a8                 tay              ;.y is xfer sp
febd  a5 00              lda  e6509       ;return bank
febf  20 2a ff           jsr  putas       ;push .a other
                                          stack
```

We've obtained from address 0 the identity of the bank we are in right now. That's bank 3, of course. We are now going to push this value to the bank 15 stack with a subroutine that uses indirect addressing again.

```
ff2a  91 ac    putas     sta  (ipoint),y  ;.a lo
ff2c  88                 dey
ff2d  60                 rts
```

The Y register is now a stack pointer for bank 15. Back to the main routine.

```
fec2  a9 04              lda  #<excrt2    ;xfer seg . . .
fec4  a2 ff              ldx  #>excrt2    ;. . . rts routn
fec6  20 24 ff           jsr  putaxs      ;.a.x to other stack
```

Now we want to put a specific return address onto the bank 15 stack. The address is $FF04, which will return us to FF05. Don't worry: bank 15 has the identical code, and will have the right thing in FF05. The subroutine that puts the address to the bank 15 stack:

```
ff24  48       putaxs    pha              ;save .a
ff25  8a                 txa
ff26  91 ac              sta  (ipoint),y  ;.x hi
ff28  88                 dey
ff29  68                 pla
ff2a  91 ac    putas     sta  (ipoint),y  ;.a lo
ff2c  88                 dey
ff2d  60                 rts
```

Maybe we should take a moment to sort out stacks here. Our stack contains two return addresses: 2004 and FFD4, plus four registers. The bank 15 stack now contains: our bank number, 3, and address FF04. Continuing the main code:

```
fec9  ba                 tsx
feca  bd 05 01           lda  $0105,x     ;.sp +5 adds lo
fecd  38                 sec
fece  e9 03              sbc  #03         ;-3 for jsr
fed0  48                 pha              ;save .a
fed1  bd 06 01           lda  $0106,x     ;hi addr
fed4  e9 00              sbc  #00
fed6  aa                 tax              ;.x hi
fed7  68                 pla              ;restore .a lo
fed8  20 24 ff           jsr  putaxs      ;push .a.x
```

Here's a bit of intricate coding. We take the address FFD4 from our own stack, subtract 3 from it to give FFD1, and put it to the bank 15 stack. Why? Because in a few moments, when we transfer to bank 15, we'll give an RTS. . . and the program will go straight to FFD2, which is the address we wanted in the first place. But there's more to do.

```
fedb  98              tya           ;xfer seg sp
fedc  38      excomm  sec
fedd  e9 04           sbc  #04      ;4 bytes .y.x.a.p
fedf  8d ff  01       sta  stackp   ;xfer seg new sp
fee2  a8              tay           ;new sp also
```

Next, we take the bank 15 stack pointer and set it ahead by four locations. Why four? We're making space for those four registers we saved long ago (remember them?). A copy of the bank 15 pointer also goes into 01FF of our bank, but that's only temporary. Now: let's move those four registers across to the stack in bank 15:

```
fee3  a2 04           ldx  #04      ;4 bytes .y.x.a.p
fee5  68      exsu10  pla
fee6  c8              iny
fee7  91 ac           sta  (ipoint),y  ;regs other stack
fee9  ca              dex
feea  d0 f9           bne exsu10
feec  ac ff  01       ldy  stackp   ;restore .y as sp
```

We messed around with Y during the move, so we need to restore its value from 01FF. Now we're getting close. We need to ask bank 15 to bring back the four registers from its stack, so we code:

```
feef  a9 2d           lda  #<expul2 ;routine to
fef1  a2 ff           ldx  #>expul2 ;. . . pull regs
fef3  20 1c ff        jsr  putaxs   ;. . . to other stack
```

A bank 15 RTS will now take it to address FF2E, which contains a routine to pull the four values.

Let's do a stack review once again. We have pulled the four registers from our bank 3 stack, so we have only addresses 2004 and FFD4 there. Come to think of it, we don't need the FFD4 any more; let's dump it:

```
fef6  68              pla
fef7  68              pla
```

What about bank 15? Its stack now contains: bank number 3; address FF04; four registers; and address FF2D.

Just before we leave this bank, we must leave the stack pointer in 01FF. Then we must pick up the correct stack pointer for bank 15:

```
fef8  ba      exgbye  tsx
fef9  8e ff  01       stx  stackp   ;save sp
fefc  98              tya           ;.y is new sp
fefd  aa              tax
fefe  9a              txs           ;new stack pointer
```

We're (finally!) ready to go. Our indirect pointer (address 1) can now become our execution pointer (address 0).

```
feff  a5 01           lda  i6509    ;xfer seg #
ff01  4c f6  ff       jmp gbye      ;good bye

fff6  85 00   gbye    sta  e6509    ;goodbye. . .
```

The instant we store into zero, we're in bank 15. Now there are things to do there.

Arrival routines

Bank 15 finds itself with the following stack setup: calling bank, 3; address FF03; address FFD1; four registers; and address FF2D. What's our next instruction?

```
fff8  60              rts
```

We dig out the FF2D from the stack, and go to address FF2E.

```
ff2e  68      expull  pla
ff2f  a8              tay           ;.y
ff30  68              pla
ff31  aa              tax           ;.x
ff32  68              pla           ;.a
ff33  28              plp           ;.p
```

Our registers are now restored, and the stack contains only: bank 3; address FF03; and address FFD1.

```
ff34  60              rts
```

Off we go to address FFD2. Isn't this where we came in? Yes, but now we're in bank 15 – with all our registers preserved – and ready to do the print job.

We've made it over. Now: can we make it back to bank 3?

Going Back

When the print routine finishes its bank 15 work, it will give an RTS. . . and we'll pick up the final address from the stack and go to FF05. The stack still contains the calling bank number, 3. First, we push five new items to the stack . . . the first one is a dummy. The SEI locks out the interrupt during the transition:

```
ff05  08      excrts  php           ;.p
ff06  08              php           ;.p
ff07  78              sei
ff08  48              pha           ;.a
ff09  8a              txa
```

```
ff0a  48          pha       ;.x
ff0b  98          tya
ff0c  48          pha       ;.y
```

Now we'll dig out the return bank (3) from the stack and place it into the indirect register:

```
ff0d  ba          tsx
ff0e  bd 06 01    lda $0106,x  ;.sp +6 is. . .
ff11  85 01       sta i6509    ; . . .return seg
ff13  20 19 ff    jsr ipinit   ;load cross sp
```

We've seen IPINIT before. It was in bank 3, but this is synchronized code, and it's exactly the same in bank 15. It picks up the "cross stack pointer" from 01FF in bank 3 – exactly where we left it. We'll stay with the main flow:

```
ff16  4c dc fe    jmp excomm
```

Hold it! We've been here before. Didn't we see EXCOMM earlier? Yup. . . but let's look at it again, briefly.

Stack report: Bank 3 (remember bank 3?) has only the return address $2004 on its stack. Bank 15, where we are now, has: bank number 3; a dummy byte; and four registers.

Would it sound familiar if I suggested that we should move the four registers over to the bank 3 stack? And would the following coding be familiar?

```
fedc  38        excomm  sec
fedd  e9 04             sbc #04      ;4 bytes .y.x.a.p
fedf  8d ff  01         sta stackp   ;xfer seg new sp
fee2  a8                tay          ;new sp also
fee3  a2 04             ldx #04      ;4 bytes .y.x.a.p
fee5  68        exsu10  pla
fee6  c8                iny
fee7  91 ac             sta (ipoint),y ;regs other stack
fee9  ca                dex
feea  d0 f9             bne exsu10
feec  ac ff  01         ldy stackp   ;restore .y as sp
```

Then we ask bank 3 to bring back the four registers from its stack, with the following familiar code:

```
feef  a9 2d       lda #<expul2 ;routine to ..
fef1  a2 ff       ldx #>expul2 ;..pull regs..
feec  20 24 ff    jsr putaxs   ;to other stack
```

Let's do a final stack review. We have pulled the four registers from our bank 15 stack, so we have only two bytes left there. And we can dump them with the old familiar code:

```
feef  68          pla
fef0  68          pla
```

What about bank 3? Its stack now contains: address 2004; four registers; and address FF2D. Can you see the conclusion coming up?

As we exit bank 15, we must – as before – leave its stack pointer in 01FF. . . for next time. And we must pick up the stack pointer for bank 3:

```
fef8  ba        exgbye  tsx
fef9  8e ff  01         stx stackp   ;save sp
fefc  98                tya          ;.y is new sp
fefd  aa                tax
fefe  9a                txs          ;new stack pointer
```

Say goodbye, bank 15:

```
feff  a5 01             lda i6509    ;xfer seg #
ff01  4c f6  ff         jmp gbye     ;good bye

fff6  85 00     gbye    sta e6509    ;goodbye. . .
```

Now bank 3 picks up again. Its stack: 2004, four registers, and FF2D. So:

```
fff8  60                rts
```

Taking us again to FF2E:

```
ff2e  68        expull  pla
ff2f  a8                tay          ;.y
ff30  68                pla
ff31  aa                tax          ;.x
ff32  68                pla          ;.a
ff33  28                plp          ;.p
```

Our registers are now restored – with the interrupt restored – and the stack contains only address 2004.

```
ff34  60                rts
```

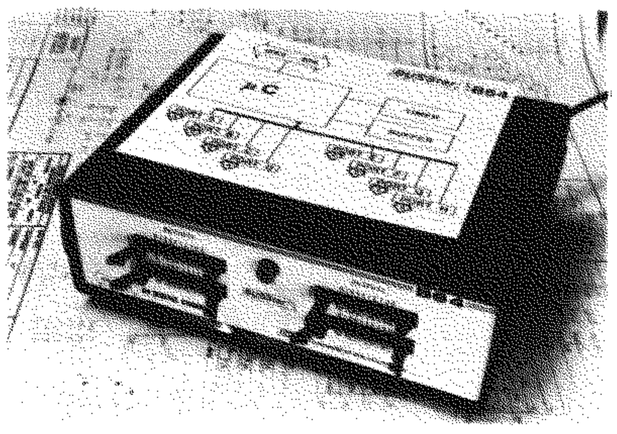
We return to address 2005. And this really is where we came in.

A Final Caution

Because code needs to by synchronized, you must have the correct ROM set in your computer. An easy way to check this is PRINT PEEK(65280). You should get a value of 1. If you don't, the above code probably won't work on your machine.

BUSSter™

LABORATORY INTERFACES FOR COMPUTERS



ANALOG AND DIGITAL INPUT/OUTOUT

The BUSSter interfaces provide analog and digital connections between any computer with an IEEE-488 or RS-232 interface and real world events. Each BUSSter product is self-contained, with its own case and power supply. They allow data acquisition while your computer is busy with other tasks. Built-in timer operates from .01 seconds to 48 hours.

- **BUSSter A64 —**
64 channel digital input module to read 64 digital signals. Built-in buffer *$495.00*

- **BUSSter B64 —**
64 channel digital output module to output 64 digital signals *$495.00*

- **BUSSter C64 —**
64 channel digital input/output module to input 32 and output 32 digital signals. Built-in buffer *$495.00*

- **BUSSTER D16 —**
16 channel analog input module to read up to 16 analog signals with 8 bit resolution (¼%). Built-in buffer *$495.00*

- **BUSSter D32 —**
32 analog channel version of the D16. . . . . *$595.00*

Add the suffix -G for IEEE-488 (GPIB) or -R for RS-232.

All prices are USA only. Prices and specifications subject to change without notice.

30 DAY TRIAL —
Purchase a BUSSter product, use it, and if you are not completely satisfied, return it within 30 days and receive a full refund.

US Dollars Quoted
$10.00 Shipping & Handling
MASTERCARD / VISA

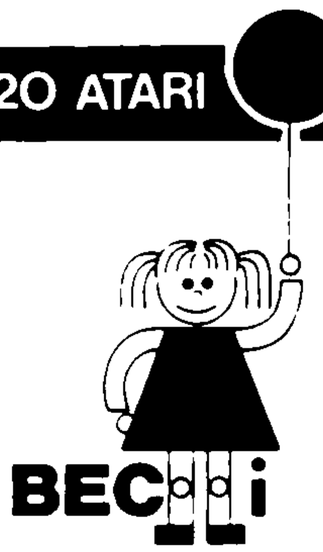
CmC

Connecticut microComputer, Inc.
INSTRUMENT DIVISION
36 Del Mar Drive
Brookfield, Ct.06804
(203) 775-4595 TWX: 710-456-0052

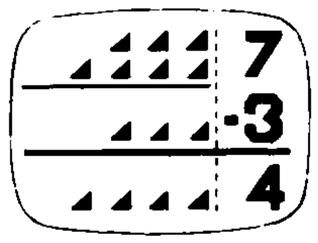
C-64 TI99/4A TIMEX VIC 20 ATARI

CHILD DEVELOPMENT SERIES

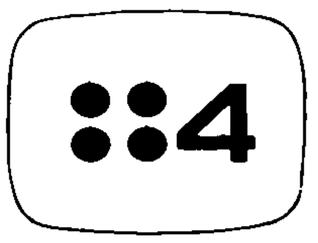
(for the 3.5K VIC and 16K ATARI)



ADD/SUB—$16.95
Displays single or multiple digits with or w/o pictures, borrows, carries, scoring, and audio/video feedback.



NUMER-BECi—$16.95
Number recognition, object counting, object grouping, and number/size/shape discrimination.



BECi is composed of professionals dedicated to providing non-trivial educational materials for the home computer. In addition to our own software, we carry a full line of evaluated hardware and software. Send $2 (refundable) for our catalog.

Send check or money order to:
BOSTON EDUCATIONAL COMPUTING, INC.
78 Dartmouth Street, Boston, MA 02116
(617) 536-5116 *MA res. add 5% tax

C64-FORTH
for the Commodore 64

FORTH SOFTWARE FOR THE COMMODORE 64

C64-FORTH (TM) for the Commodore 64 - $99.95
- Fig Forth-79 implementation with extensions
- Full feature screen editor and macro assembler
- Trace feature for easy debugging
- 320x200, 2 color bit mapped graphics
- 16 color sprite and character graphics
- Compatible with VIC peripherals including disks, data set, modem, printer and cartridges
- Extensive 144 page manual with examples and application screens
- "SAVETURNKEY" normally allows application program distribution without licensing or royalties

C64-XTEND (TM) FORTH Extension for **C64-FORTH** - $59.95
(Requires original **C64-FORTH** copy)
- Fully compatible floating point package including arithmetic, relational, logical and transcendental functions
- Floating point range of 1E+38 to 2E-39
- String extensions including LEFT$, RIGHT$, and MID$
- BCD functions for 10 digit numbers including multiply, divide, and percentage. BCD numbers may by used for DOLLAR.CENTS calculations without the round-off error inherent in BASIC real numbers.
- Special words are provided for inputting and outputting DOLLAR.CENTS values
- Detailed manual with examples and applications screens

(Commodore 64 is a trademark of Commodore)

TO ORDER - Specify disk or cassette version
- Check, money order, bank card, COD's add $1.50
- Add $4.00 postage and handling in USA and Canada
- Mass. orders add 5% sales tax
- Foreign orders add 20% shipping and handling
- Dealer inquiries welcome

PERFORMANCE MICRO PRODUCTS

MasterCard

770 Dedham Street. S-2
Canton, MA 02021
(617) 828-1209

VISA

SEND FOR OUR CATALOG OF COMMODORE 64–VIC 20 SOFTWARE. Over 150 different pieces of software and accessories for the CBM 64. With full descriptions and screen pictures. **Send $2.00 postage and handling.**

VIC - 20 SOFTWARE

MICRO SPEC

DATA BASE MGR.	$49.95
HOME INVENTORY	15.95
CHECKBOOK MANAGER	39.95
DATA MANAGER	17.95

HES

HES WRITER	$31.95
6502 PROF. DEV. SYS.	26.95
GRIDRUNNER	31.95
SHAMUS	31.95

TOTAL

TIME MANAGEMENT	$24.95
MAILING LIST & LABELS	17.95
RESEARCH ASST.	24.95

64 SOFTWARE

CALC RESULT.....$139.95
3 Dimensional Electronic Spreadsheet with: built-in graphics, on-line help menus, full arithmetic functions, Boolean operations, and split screen/split window features that allow you to view up to 4 different pages at the same time – EXCELLENT –

SYSRES.....$79.95
Programmer's utility package gives you AUTO-NUMBER, RE-NUMBER, MERGE, SEARCH AND REPLACE, TRACE, DUMP, FULL DOS SUPPORT, and even a non-destructive disk directory
– MANY MORE FEATURES –

PAPERCLIP.....$125.00
Professional Word Processor. Full Screen Editing, line length up to 126 char. using horizontal scrolling, supports any printer, even works on 80-column board. This program does so much we can't list it – call for more information.

M-FILE.....$99.95
A truly relational data base manager. Up to 1200 records in a file, all arithmetic operations, and super report generation using search and sort.

SPELLMASTER.....$89.95
Proofreading software for the C-64! Now with the PaperClip and SPELLMASTER, you can have it all in word processing convenience and power. Expandable and easy.

DATA 20 VIDEO PAK 80.....$164.95
Gives you a full 80 column screen display, plus terminal emulation software (use with monochrome monitor only). Compatible with PaperClip WP.

OTHER SERIOUS SOFTWARE

VISICALC	$219.95
PET/SPEED BASIC COMPILER	149.95
QUICK BROWN FOX	54.95
MICRO SPEC CHECKBOOK	39.95
MICRO SPEC MAILING LIST	44.95
MICRO SPEC INVENTORY	79.95
MICRO SPEC GEN/LEDGER	79.95
MICRO SPEC PAYROLL	79.95
MICRO SPEC DATA BASE MGR.	74.95
MINI JINI DATA BASE (CART.)	79.95
BUSICALC	69.95
'64 TERMINAL	24.95

GAMES

INFOCOM

STARCROSS	$31.95
ZORK I	31.95
ZORK II	31.95
ZORK III	31.95
DEADLINE	41.95
SUSPENDED	41.95

EPYX/AUTOMATED SIMULATIONS

TEMPLE OF APSHAI	$34.95
UPPER REACHES OF APSHAI	17.95
CURSE OF RA	17.95
JUMPMAN	31.95

SIERRA ON LINE

FROGGER	$29.95
JAWBREAKER	21.95

ACCESS SOFTWARE

NEUTRAL ZONE	$29.95
SPRITE MASTER	28.95

TIME WORKS

WALL STREET	$21.95
ROBBERS OF THE LOST TOMB	21.95

COMM*DATA

CETROPODS (D/C)	$16.95
PAKACUDA (D/C)	16.95
SKETCH & PAINT	13.95

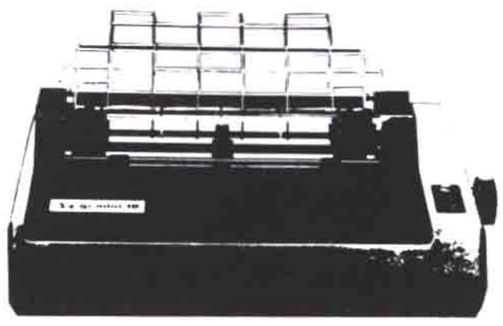
HARDWARE & ACCESSORIES

SPECIAL OF THE MONTH

GEMINI 10X $399.95*

WITH CARDCO INTERFACE

FOR VIC-20
OR COMMODORE 64
120 c.p.s., BIDIRECTIONAL, DOT ADDRESSABLE GRAPHICS, ITALICS, 4 DIFFERENT PITCHES



COMMODORE 64	$249.95 *
1541 DISK DRIVE	259.95 *
1525 PRINTER	239.95 *
1701 COLOR MONITOR	279.95 *
1520 4 COLOR PLOTTER	199.95
1600 MODEM	99.95
1530 DATASETTE	60.00
COMMODORE 64 REF. MANUAL	17.95
GEMINI 10 PRINTER	349.95 *
SCM TPI PRINTER	599.95 *
CARDCO CARD INTERFACE	69.95
CARDCO CASSETTE INTERFACE	34.95
CARDCO LIGHT PEN	24.95

***PLEASE INCLUDE AN ADDITIONAL $5.00 FOR SHIPPING.**

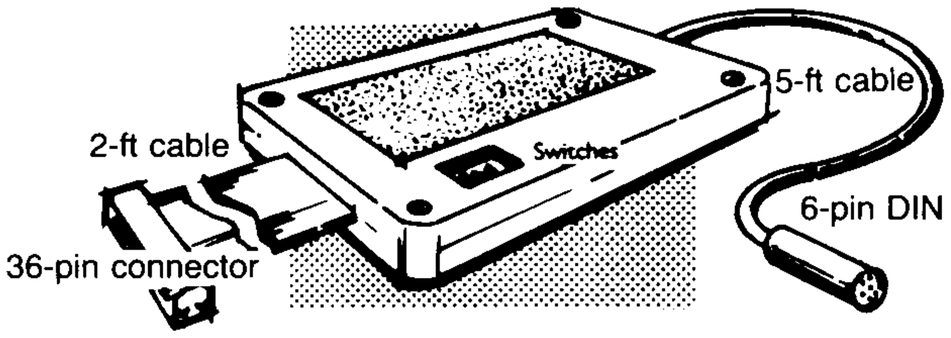
SOFTWARE INTL

560 N. Mountain Ave., Suite L • Upland, CA 91786 • (714) 981-5925

TO ORDER: Send check or money order plus $3.00 ($8.00 on indicated items) postage and handling. California residents add 6% sales tax. VISA and MasterCard are accepted. Give account number and expiration date. All orders shipped UPS Ground. Next-Day and Second-Day Service available. Call for prices. All prices in U.S. dollars.

MW-302: VIC-20/64

Parallel Printer Interface.



Works with all centronics type parallel matrix & letter printers and plotters—Epson, C.Itoh, Okidata, Nec, Gemini 10, TP-I Smith Corona, and most others. Hardware driven; works off the serial port. Quality construction: Steel DIN connectors & Shielded cables. Has these switch selectable options: Device 4, 5, 6 or 7; ASCII or PET ASCII; 7-bit or 8-bit output; upper & lower case or upper only. Recommended by PROFESSIONAL SOFTWARE for WordPro 3 Plus for the 64, and by City Software for PaperClip.

MW-302 . . . . Canadian $189.95

Micro World Electronix, Inc.
3333 S. Wadsworth Blvd. #C105, Lakewood, CO 80227
(303) 987-2671

CANADIAN DEALERS

ALBERTA

Computer Shop of Calgary
3515 18th St. S.W.
Calgary, T2T 4T9
(403) 243-4356

Hindson Computer Systems, Ltd.
7144 Fisher St. S.E.
Calgary, T2H 0W5
(403) 252-9576

TJB Micro Systems, Ltd.
10991 124th St.
Edmonton, T5M 0H9
(403) 433-3161

BRITISH COLUMBIA

Conti Electronics
7204 Main Street
Vancouver, V5X 3Y4
(604) 324-0505

ONTARIO

MGI Computer Corp.
1501 Carling Ave.
Ottawa, T1Z 7M1
(613) 722-1000

Richvale Telecommunications
10610 Bayview (Bayview Plaza)
Richmond Hill, L4C 3N8
(416)884-4165

SASKATCHEWAN

Micro Shack of West Canada
607 45th St. West
Saskatoon, S7L 5W5
(306) 244-6909

PRO·LINE SOFTWARE

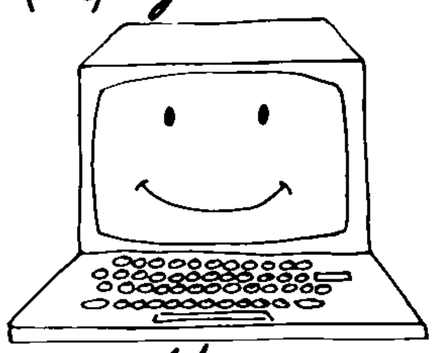
A CANADIAN COMPANY

designing, developing, manufacturing, publishing and distributing microcomputer software

DEALER ENQUIRIES WELCOME
AUTHOR'S SUBMISSIONS INVITED
CALL OR WRITE

(416) 273-6350
PRO·LINE SOFTWARE
755 THE QUEENSWAY EAST, UNIT 8.
MISSISSAUGA, ONTARIO L4Y 4C5

(M)agreeable™



software

STOCK HELPER™

Commodore 64 and VIC-20

Stock HELPER is a tool to maintain a history of stock prices and market indicators on diskette, to display charts, and to calculate moving averages. Stock HELPER was designed and written by a "weekend investor" for other weekend investors.

Stock HELPER is available on diskette for:

Commodore 64	$30.00	($37.00 Canadian)
VIC-20 (16K)	$27.00	($33.25 Canadian)

plus $1.25 shipping ($1.55 Canadian)

The VIC-20 version only charts 26 bi-weekly periods rather than 52 weekly periods.

(M)agreeable software, inc.
5925 Magnolia Lane • Plymouth, MN 55442
(612) 559-1108

(M)agreeable and HELPER are trademarks of (M)agreeable software, inc.
Commodore 64 and VIC-20 are trademarks of Commodore Electronics Ltd.

If our word processing software is so great, why are we giving it away?

It's our way of introducing you to DATA 20.

As the leader in price/performance peripherals for Commodore 64™ and VIC 20,™ we want you as a loyal customer. So to encourage you to sample our add-ons, we're giving away our WORD MANAGER software when you purchase any DATA 20 DISPLAY MANAGER or VIDEO PAK to expand your Commodore to 80 columns.

What's so great about WORD MANAGER?

The table tells the tale. This DATA 20 exclusive gives you powerful features found only on the most expensive word processing systems including mail merge, block move, search and replace. And we've written our word processing in machine language for fast execution and low memory requirements.

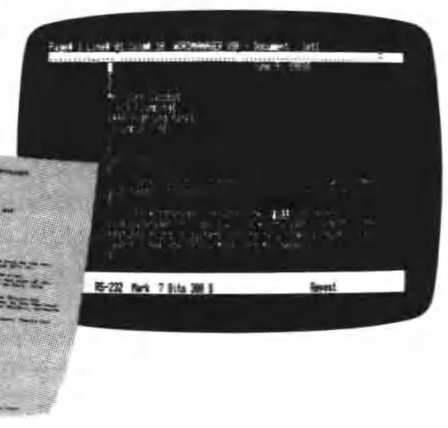
Compare the Top Three	Mail Merge	Feature Strip	Works with Any Printer*	Tape & Disk Capabilities	Full Screen Editing	80-Column Format
WORD MANAGER	•	•	•	•	•	•
WordPro®	—	—	—	—	•	—
Quick Brown Fox™	—	—	•	—	—	•

*Most standard serial or parallel printers require interface, not supplied.

Best of all, WORD MANAGER is exceptionally easy to use. A self-adhesive feature strip for function keys makes most commands one-key simple, eliminating awkward prompts and menus.

What you see is what you print.

With our professional 80-column format, your printed product duplicates screen output precisely. You see centering, left justification and other features as they will print. So you'll save time, save paper, and get the exact results you want!



WORD MANAGER is really something—for nothing! Get it free with your DISPLAY MANAGER or VIDEO PAK from DATA 20. See your dealer or write: DATA 20 CORPORATION, 23011 Moulton Parkway, Suite B10, Laguna Hills, CA 92653.

DATA 20 CORPORATION

Price/Performance Peripherals

Commodore 64 and VIC 20 are trademarks of Commodore Electronics, Ltd.
Quick Brown Fox is a trademark of Quick Brown Fox.
WordPro is a registered trademark of Professional Software, Inc.

PET / CBM™

SOFTWARE SELECT!

8032 DISPLAY OR **4032 DISPLAY**

FROM THE KEYBOARD OR PROGRAM
NOW RUN WORD PRO 3 OR WORD PRO 4

FROM THE SAME MACHINE

Available for either 4000 or 8000 Series

ALSO:
For **2001 / 3000** Series Computers

Operate these Models in a Full **8032** Like Display For Word Pro 4*
and all other 80 Column Software
All installation instructions included.

EXECOM CORP.

1901 Polaris Ave.
Racine, WI 53404
Ph. 414-632-1004

PET/CBM a trademark of Commodore Business Machines
*trademark of Professional Software, Inc.

TYPRO

DATA MANAGER & WORD PROCESSOR

For COMMODORE 8032 Computer — 8050/4040 Dr.

NOW AVAILABLE FOR THE COMMODORE 64®

DATA MANAGER

Number of records is only limited by your disk capacity. Up to 50 fields per record. Maximum of 75 characters per field. User formatted. Screen editing. Sort and search feature. Pattern match search. Selective field printing and formatting. Form letter addressing. Mailing list and mailing label printing. Format for fanfold Rolodex and index card printing.

WORD PROCESSOR

Screen editing. Automatic line length set. Add, move or delete text. Global edit. Page numbering and titling. Form letter addressing. File append for printing.

BOTH PROGRAMS ABOVE, ONLY $89.00

* Commodore 64 is a trademark of Commodore Electronics, Ltd.

Also for Commodore 64 and 8032
AMORTIZATION SCHEDULE — $30.00 (Disk)
INVENTORY MANAGEMENT — $55.00 (Disk)

All software is fully supported for updates and revisions for up to six months after purchase.
Specify Computer model number and Disk model number.

INPUT SYSTEMS, INC.

25101 S.W. 194 Ave. Homestead, FL 33031 (305) 245-3141
DEALER INQUIRIES INVITED.

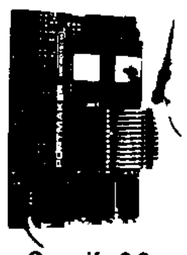
Now for the "64"

STCP — 300/1200 Baud

Standard Terminal Communications Package

PFO IOD OOA CP<D1>D2 BELL = 12:30:00 10:14:36

Don't settle for non-standard Communications Protocol! Access Micro Net, Source, Bulletin Boards, Local Mainframe, etc.



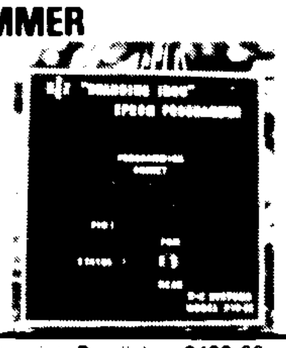
- Complete Package — Includes RS232 Interface Board and software (does not include modem)
- Communicates in Industry Standard ASCII
- Upload/Download to/from Disk
- Automatic File Translation
- Can be controlled from keyboard or user supplied basic or machine language program

Specify 3.0 or 4.0 ROMS or 8032 Commodore Computer 4040 or 8050 or PEDISK II Disk or CBM64 on 1541.

Price: $129.95

ATARI AND PET EPROM PROGRAMMER

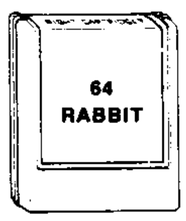
Programs 2716 and 2532 EPROMs. Includes hardware and software. PET = $75.00 – ATARI (includes sophisticated machine language monitor) = $119.95



Prowriter Printer - Excellent dot matrix print. Parallel = $489.00
Serial = $600.00. IEEE = $589.00

NEW

VIC RABBIT CARTRIDGE AND CBM 64 RABBIT CARTRIDGE



NEW FEATURE! DATA FILES!

"High Speed Cassette Load and Save!"

$39.95 (includes cartridge and manual)

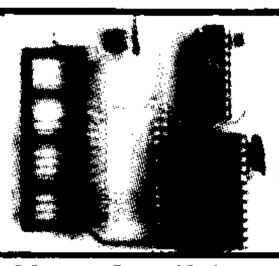
for VIC

Don't waste your Life away waiting to LOAD and SAVE programs on Cassette Deck.

Load or Save 8K in approximately 30 seconds! Try it—your Un-Rabbitized VIC or 64 takes almost 3 minutes. It's not only fast but VERY RELIABLE.

Almost as fast as 1541 Disk Drive! Don't be foolish — Why buy the disk when you can get the Rabbit for much, much less!

Allows one to APPEND Basic Programs!
Easy to install — just plugs in.
Expansion Connector on rear of the VIC Rabbit.
Works with or without Expansion Memory.
Works with VIC or 64 Cassette Deck.
12 Commands provide other neat features.
Fast Data Files - two data file modes.
Also Available for 2001, 4001, and 8032.



TRAP 65

TRAP 65 is a hardware device that plugs into your 6502's socket. Prevents execution of unimplemented opcodes and provides capability to extend the machines' instruction set.
For PET/APPLE/SYM.
Reduced from $149.95 to $69.95

DC Hayes Smart Modem = $235.00
DC Hayes Micro Modem II = $289.00

Rana Disk Drive - 375
4 Drive Controller - 114

More than just an Assembler/Editor!

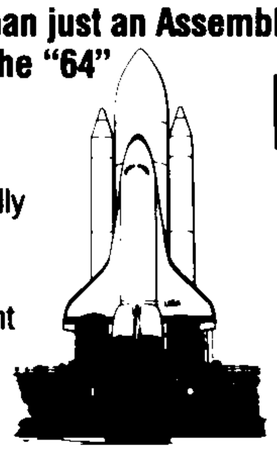
Now for the "64"

MAE

for PET APPLE ATARI

~~$169.95~~ New Price $99.95

It's a Professionally Designed Software Development System



Blast off with the software used on the space shuttle project!

- Designed to improve Programmer Productivity.
- Similar syntax and commands — No need to relearn peculiar syntaxes and commands when you go from PET to APPLE to ATARI.
- Coresident Assembler/Editor — No need to load the Editor then the Assembler then the Editor, etc.
- Also includes Word Processor, Relocating Loader, and much more.
- Options: EPROM Programmer, unimplemented opcode circuitry.
- STILL NOT CONVINCED: Send for free spec sheet!

5¼ INCH SOFT SECTORED DISKETTES

Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers. $22.50/10 or $44.50/20

EPROMS 2716 = $4.50 2532 = $7.50
Over 40 Commodore Programs by Baker (on 4040) = $25.00

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA
MasterCard

Commodore 64 and VIC-20

Graphics

"ELF" from Picture Library

DRAW with your joystick!

Now, you can create **high-resolution pictures** on-screen with your joystick as a "pen." Design critters, objects, pie-charts — whatever your imagination wishes! **SAVE your creations to tape or disk, and PRINT them on a VIC printer.** Educational and fun!

Draw narrow or wide lines, curvy or straight; set points; add captions; create background patterns; change picture, background, and border colors; reverse colors for a negative; even connect dots with straight lines automatically! **You control every dot on the screen.**

A large "Picture Library" is included to get you started, plus a 20-page instruction manual. Joystick required. VIC printer and disk drive optional. On cassette.

"FLY" from Picture Library

For the Commodore 64:
'64 Panorama $29.95

For the VIC-20:
VIC-PICS $29.95

(Full features need 8K mem exp; reduced version included for unexpanded VIC.)

PRINT (the unprintable)... with Smart ASCII Plus–$59.95

Now, print the unprintable **Commodore graphics** on your dot-addressable printer* with Smart ASCII Plus. This **powerful, low-cost software interface** converts your USER PORT into a fast, intelligent port for "Centronics" protocol parallel printers.

Six flexible print modes: GRAPHICS, TRANSLATE, DaisyTRANSLATE, CBM ASCII, True ASCII, PIPELINE. GRAPHICS mode creates actual VIC/64 keyboard graphics. TRANSLATE converts normally unprintable control-codes into text: (CLR), (RVS), (BLU), etc., with an extended mode for Daisywheel printers. Convenient set-up menu and simplified operation.

Smart ASCII Plus is compatible with most application programs: WordPro 3+, EasyScript, Quick Brown Fox (for the VIC), Writer's Assistant, etc.

Complete with connecting cable for printer and instruction manual. On cassette. Copy to your disk for quick loading. (Upgrades available for original Smart ASCII owners.)

*Requires dot-addressable printer such as: Epson FX-80 or MX-80/100 with Graftrax; Okidata Microline 84; C. Itoh Prowriter 1 & 2; Star Micronics Gemini-10 or 15.

(816) 333-7200

Send for a free brochure.

MAIL ORDER: Add $1.50 shipping and handling ($3.50 for C.O.D.); VISA/Mastercard accepted (card# and exp. date). MO residents add 5.625% sales tax. Foreign orders payable U.S.$, U.S. Bank ONLY; add $5 shp/hndlg.

MIDWEST MICRO inc.

311 WEST 72nd ST. • KANSAS CITY • MO • 64114

VIC 20 and Commodore 64 are trademarks of Commodore Electronics, Ltd.

NEW

Basic Utility for the Commodore 64

POWER 64

- **easy to learn**
- **easy to use**
- **program faster and more efficiently with better results**
- **MOREPOWER included FREE**

Powerful Programmer's Utility
by Brad Templeton
Manual by Jim Butterfield

$99.95 from your local Commodore dealer.
For your nearest dealer call:
(416) 273-6350

PRO·LINE SOFTWARE
755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO L4Y 4C5

CURSOR

For your Commodore 64

For only $12.95 each, our **CURSOR 64** tapes are your best buy for the Commodore 64. They take advantage of the color, sound, and sprites that make the 64 such a delight to use. Most of our packages include three excellent Basic programs on one cassette tape. The programs are not copy protected, so you can look at the source code, and learn how to make the 64 do its tricks.

We don't have room to describe all 25 of our **CURSOR 64** programs here. As a sample, you may want to order tape 64-5 with the exciting **Godzilla** program. You'll be challenged as you try to save Tokyo from from the rampaging Godzilla. Or try tape 64-3 with the popular **Miser** text adventure that will take you hours to solve (even if you cheat and read the program source).

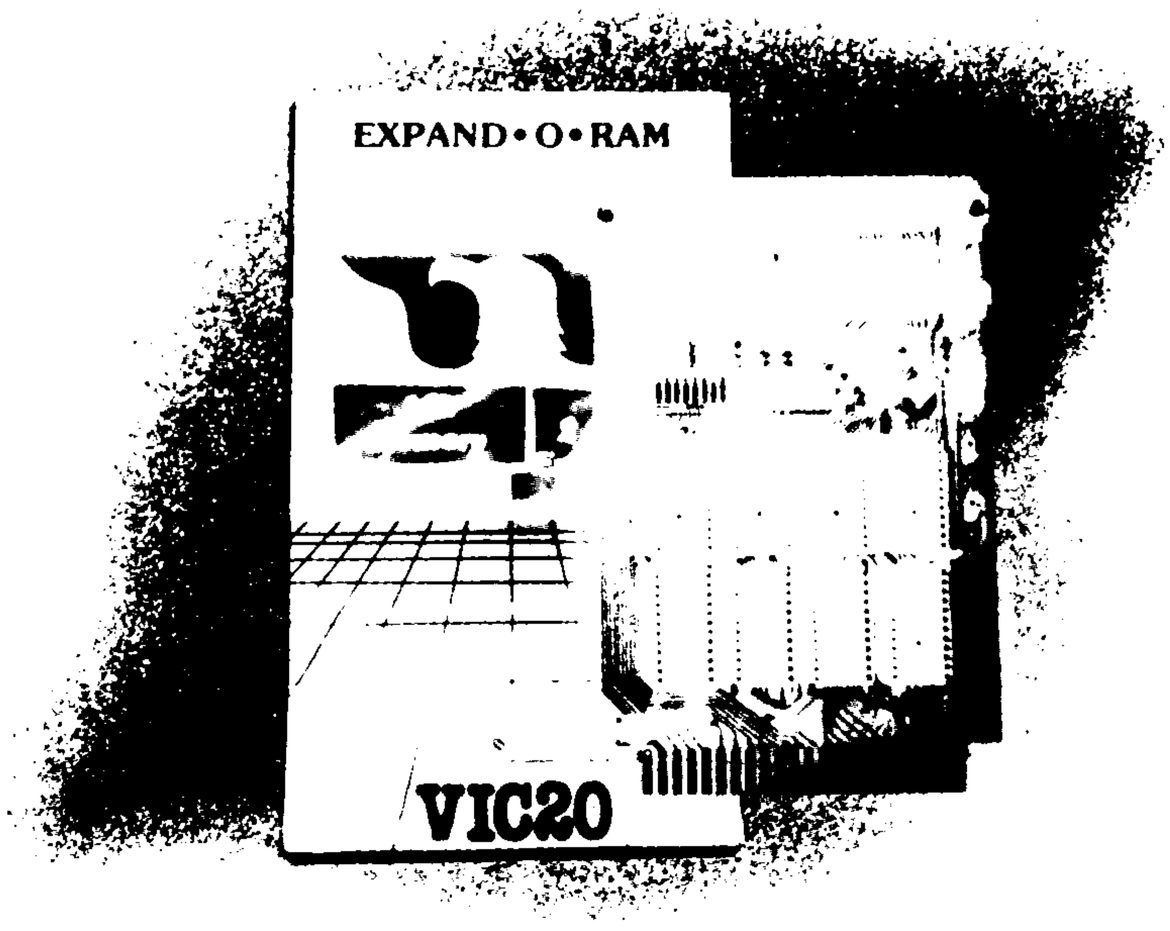
We have super programs for the VIC 20, such as **Dungeon** ($12.95), a visual adventure for 16K VICs. Our **VIXEL** programs are also popular with VIC owners. And, we still sell all 30 of the original **CURSOR** cassettes for the original PET and CBM.

Call or write for a catalog today. Be sure and tell us whether you have a 64, a VIC, or a PET. We welcome credit cards, and ship most orders the same day they are received. Dealer inquiries invited.

CURSOR 64, Box 6905
Santa Barbara, CA 93110
805-683-1585

Technological Innovations for the VIC 20™ and Commodore 64™

Designed by RAK-Ware



EXPAND-O-RAM VIC20

EXPAND-O-RAM is truly a multifunction memory expansion board that provides 2 switch selectable slots to double the capacity of your computer. In addition to these important features, a RESET button has been provided to allow you to regain control of a "hung" or unstoppable program. A bank of mini-switches control memory addressing on the board so that you can use EXPAND-O-RAM as a useful tool for cartridge development and debugging. A write protect function is even provided so that you can simulate a ROM environment or investigate a previously programmed ROM Cartridge Pack. A quick summary of the features are as follows:

A) 16K of additional Memory.
B) Compatability with additional Memory & Games Cartridges.
C) Two Additional Switch Selectable Slots.
D) Reset Switch for warm starts.
E) Slide Switch for easy selection of Cartridges or memory packs.
F) Switch selectable Memory Map selections of the 16K memory.
G) Use as a Program and Cartridge development tool.
H) Write protection of memory area for cartridge simulation and investigation.

ALL THIS FOR ONLY.....$119.00

UNIVERSAL TAPE INTERFACE

No need to purchase an expensive data recorder for your VIC 20™ or Commodore 64™. The Tymac Universal Tape Interface and Duplicator will allow you to load, save and even duplicate* your programs and data cassettes with virtually any portable home tape recorder. To insure positive LOADS and SAVES, a special audio enhancement circuit is provided in the interface. Three LED indicators monitor the status at all times and provide a visual indication of loads and saves. A parity switch will also insure that all types of data tapes can be successfully loaded. Operator controlled action of the recorder is another desirable feature. Finally, you can easily make back-up tapes without the need of loading your software back into the computer. A unique switch function will allow you to make direct recordings to another tape recorder. A great duplication device for the user who may want to start a software business at home. The TYMAC UNIVERSAL TAPE INTERFACE AND DUPLICATOR is way ahead of the competitors in features and quality.

ALL THIS FOR ONLY.....$49.00

*NOTE: Duplication requires the use of Two Standard tape recorders.

MICRO WARE
DISTRIBUTING INC
1342 B Rt. 23, Butler, NJ 07405
201-838-9027

MasterCard VISA

Dealer and Distributor Inquiries Invited

NOTE We solicit hardware and software items for the VIC 20* and CBM 64* Royalties, license fees, or outright purchases can be negotiated CBM 64* & VIC 20* are Registered Trademarks of Commodore Business Machines Inc

Software first, computer second.

Workhorse solutions for tough questions.

OK. You're ready to buy a computer. Here's how to make an intelligent business decision.

Decide on your software first. No computer is better than the software that runs the operation. No software is better than **Southern Solutions'.** We have real business accounting and record keeping software that is right for today's business world. It runs on the best line of computers available today: Commodore including the exciting new Commodore 64™.

Compare our software solutions with others:

FileGuard™ protects your files from loss even by power failure.

SuperMath™ meets your needs as your business expands. Our software with SuperMath will handle numbers up to $1 billion. Most micros stop at far less.

You can design your Balance Sheet, P&L, Budget Analysis, etc.

Complete Systems or Individual Modules handle general ledger, accounts receivable and payable, billing, payroll, mailing lists, oil accounting, pharmacy management, encumbrance accounting, etc.

Our software uses practically any printer and grows as your needs expand.

Real business software for real business computers.
Capability you need at prices you can afford.

Your professional computer dealer can help you make a computer become a productivity tool. For a demonstration, visit

Distributed in Canada by: Canadian Micro Distributors Ltd*
500 Steeles Avenue
Milton, Ontario
L9T 3P7

OFFICIALLY APPROVED BY COMMODORE

Now For The Commodore 64

W E
Southern Solutions

PO. Box 'P'
McKinney, Texas
75069
(214) 542 – 0278

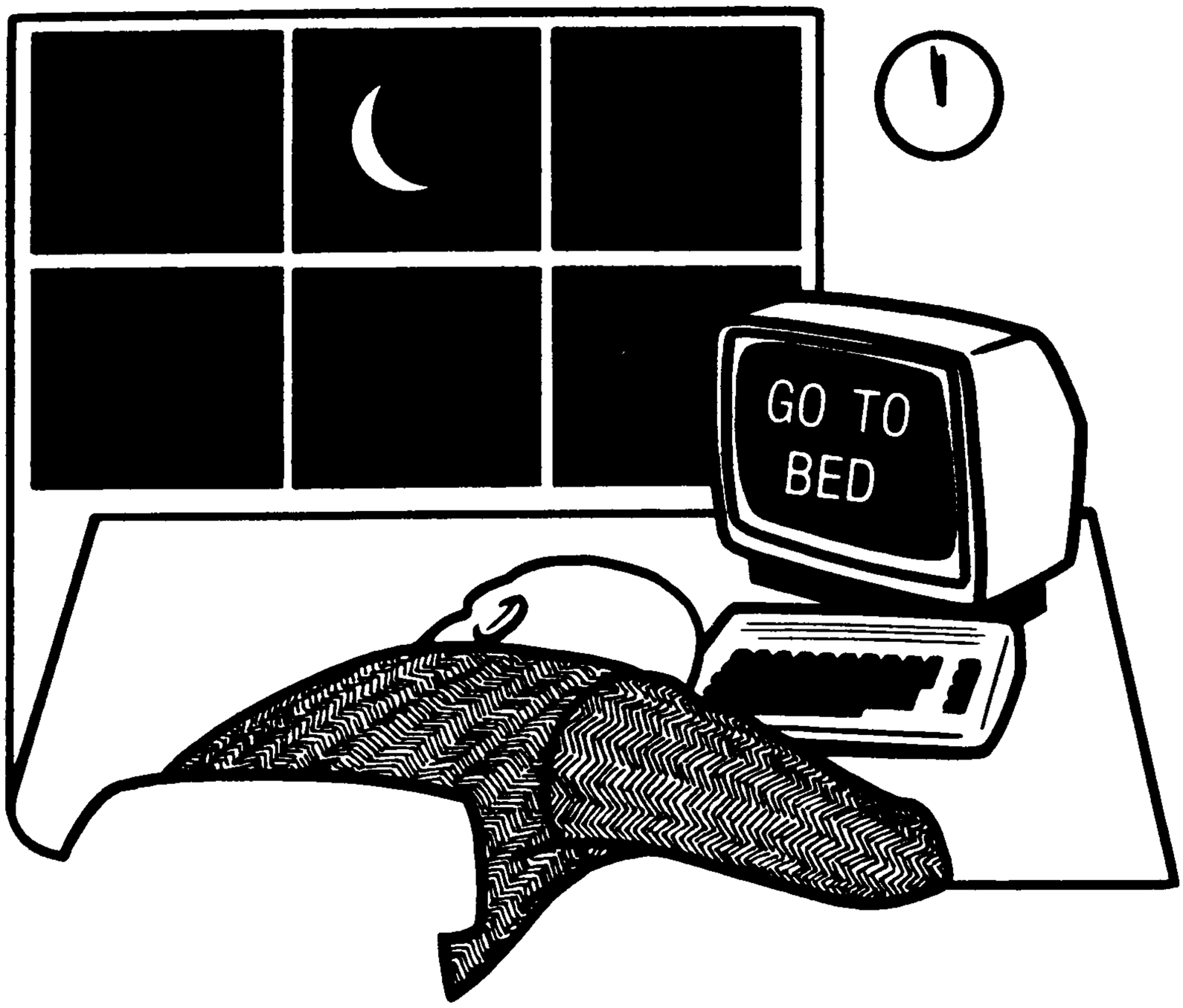
* A Division of BMB CompuScience Ltd.

''Commodore 64 is a trademark of Commodore Business Machines, Inc.''

The MIDNITE SOFTWARE GAZETTE

The PAPER

Five years of service to the PET community.



The Independent U.S. magazine for users of Commodore brand computers.

EDITORS: Jim and Ellen Strasma
Sample issue free on request, from:

635 MAPLE □ MT. ZION, IL 62549 USA

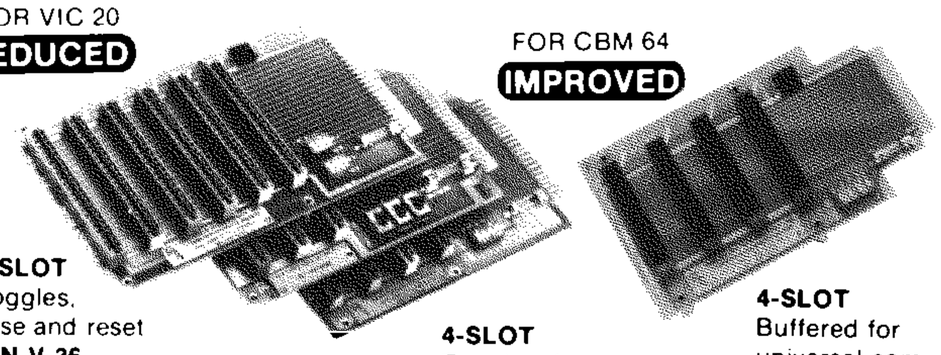
VIC 20™ and CBM 64™ EXPANDER BOARDS

FOR VIC 20

REDUCED

FOR CBM 64

IMPROVED



6-SLOT
Toggles, fuse and reset
P/N V-36
$69.95

6-SLOT with 3-ft. ribbon cable
P/N V-46
$89.95

3-SLOT
Slide switches and fuse
P/N V-23
$49.95

4-SLOT
Toggles, fuse and reset
P/N V-24
$59.95

4-SLOT
Buffered for universal compatibility. Toggles, fuse and reset.
$69.95

NEW VIC 20™ MEMORY

A multipurpose board with sockets for three 8K blocks of static RAM or EPROM

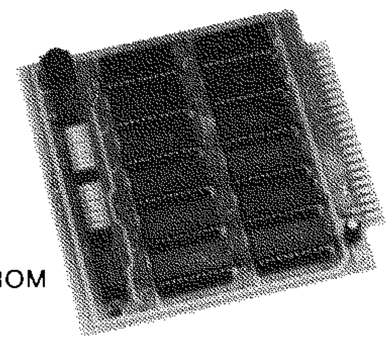
Expands VIC 20 memory to 29K when equipped with 24K RAM

System reset button

Control switches for 8K memory blocks

6116 RAM or 2716 EPROM

90-day warranty



Gold connector

Sockets for all ICs

Sold with 8, 16 or 24K RAM

P/N VME-1
with 8K RAM **$79.95**
with 16K RAM **$99.95**
with 24K RAM **$119.95**

shown without cover

MONITOR/AUDIO CABLE

Connects VIC 20 or CBM 64 to audio amplifier and TV monitor

PRECISION TECHNOLOGY, INC.
COMPUTER PRODUCTS DIVISION
P.O. BOX 15454
SALT LAKE CITY, UTAH 84115
(801) 487-6266

Color 64 or VIC **P/N MC-2 $12.95**
B & W 64 only **P/N MC-3 $12.95**

See your dealer or place your order directly

VISA - M/C - CHECK - COD

TM-Trademark of Commodore Electronics Limited

NEW

Assembler for the Commodore 64

PAL64

- **easy to learn**
- **easy to use**
- **fast**
- **comprehensive manual**

Personal assembly language
by Brad Templeton
also available for the Commodore
4,000 - 8,000 - 9,000 series

$99.95 from your local Commodore dealer.

For your nearest dealer call:

(416) 273-6350

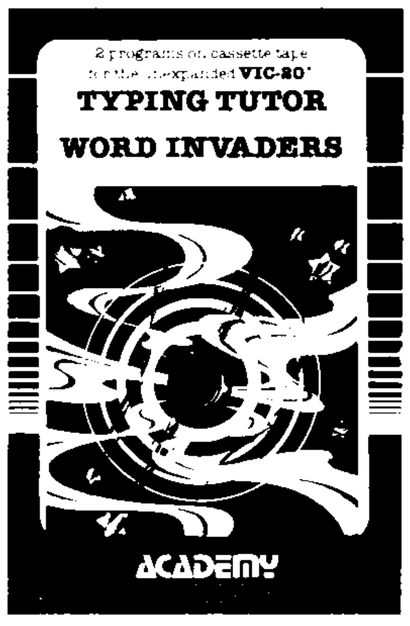
PRO·LINE SOFTWARE

755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO L4Y 4C5

FOR THE VIC 20™

JOIN THE COMPUTER REVOLUTION WITH A MASTERY OF THE KEYBOARD!

In the age of the computer, *everyone* from the school child to the Chairman of the Board should be at home at the computer keyboard. Soon there will be a computer terminal on every desk and in every home. Learn how to use it right ...and have some fun at the same time!



Rated THE BEST educational program for the VIC 20™ by Creative Computing Magazine

TYPING TUTOR PLUS WORD INVADERS — $21.95

(2 programs on one cassette tape for the unexpanded VIC 20™)

Typing Tutor plus Word Invaders makes learning the keyboard easy and fun! Typing Tutor teaches the keyboard in easy steps. Word Invaders makes typing practice an entertaining game. Highly praised by customers:

"Typing Tutor is great!", "Fantastic", "Excellent", High quality", "A source of great joy and learning for our children", "Even my little sister likes it", "Word Invaders is sensational!"

Customer comment says it all . . .

". . . and it was everything you advertised it would be. In three weeks, my 13 year old son, who had never typed before, was typing 35 w.p.m. I had improved my typing speed 15 w.p.m. and my husband was able to keep up with his college typing class by practicing at home."

FOR THE COMMODORE 64™

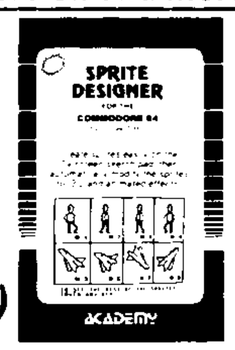
SPECIAL VERSION OF TYPING TUTOR PLUS WORD INVADERS NOW AVAILABLE FOR THE COMMODORE 64™ ... $21.95 *(Tape)* $24.95 *(Disk)*

All of the features of the VIC 20™ Version and more

SPECIAL!

SPRITE DESIGNER

by Dr. Lee T. Hill



$16.95 *(Tape)* **$21.95** *(Disk)*

Create and then transform sprites automatically. We have the other sprite making programs, but this is the one *we use* to make sprites. The automatic transformations are great!

VISA

Shipping and handling $1.00 per order. California residents add 6% sales tax. VISA and Mastercard orders must include full name as shown on card, card number, and expiration date. Free catalog sent with order and on request.

MasterCard

ACADEMY SOFTWARE

P.O. Box 9403, San Rafael, CA 94912 (415) 499-0850

Programmers: Write to our New Program Manager concerning any exceptional VIC 20™ or Commodore 64™ game or other program you have developed.

MAILPRO
STEVE PUNTER'S DATA ORGANIZER AND MAILING LIST PROGRAM FOR COMMODORE 64™

COMPARE THESE FEATURES:

- fast file definition
- easy updating
- rapid printing with total format and record selection control
- WORDPRO compatible
- up to 4000 records on 1541

MAILPRO 64.....$129.95

Also available for COMMODORE 8032 . . . $179.95

Call for the name of your local dealer:

PRO·LINE SOFTWARE

PRO-LINE SOFTWARE LTD.

(416) 273-6350

755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO CANADA, L4Y 4C5

Intelligent Software for the Commodore computers

Catalog 5/1/83

My line of programs (such as it is) consists of the following products. All are written for Commodore computers; any of my programs will load and run without modification in the entire line (including older PET's).

1. Word Processor; $25. It includes the following features: VERY fast file routines, including a disk file catalog; automatic form handling on tractor- or friction-feed printers; fully imbedded margin, justification, spacing, formatting, and paging controls; block commands and error-trapping in editing mode; and a spool routine (formatted output to disk for later mass printing). I believe W/P is the most thoroughly tested, user-oriented word processor available at this time at anywhere near the price. for any machine. Requires a minimum of 10k of memory (8k expansion on VIC), and a printer.

2. Copycalc; $20 ($15 if ordered with another program). Copycalc is a simplified version of the "electronic spreadsheets" that are becoming extremely popular for use on personal computers. It allows the user to set up a visible grid of numbers on the screen, and use the screen-editor to make changes in the grid, with the totals reflecting the changes. Requires 6k RAM (3k expansion on VIC); smaller version available for unexpanded VIC.

3. Baseball Manager; $30. This program maintains complete batting statistics for a baseball or softball league of up to 250 players. It generates reports on a player, team, or the entire league (including standings). It requires a minimum 10k of RAM; a printer is suggested but not required.

4. Inventory; $30. A general-purpose perpetual inventory control program. It produces a variety of reports, including order forms; multiple vendors are supported. Requires 10k of RAM; a printer is suggested.

All programs: support cassette and disk files and the CBM printers (easily modifiable to other printers), come on cassette, and include documentation. Prices include shipping; Calif. residents add 6%. All programs are copyrighted by the author; those rights will be enforced. Programs available from:

William Robbins, Box 3745, San Rafael, CA 94912

NEW

MPI INTRODUCES SUPER ACTION MEMORY EXPANDER BOARD FOR VIC 20*

Adds 24K and 3 Expansion Slots With Switches and Fuse $109.95

(Expandable To 35K By Simply Adding Memory Chips and Switches)

INTRODUCTORY PRICE

$109.95

Transfer Cartridges To Tape

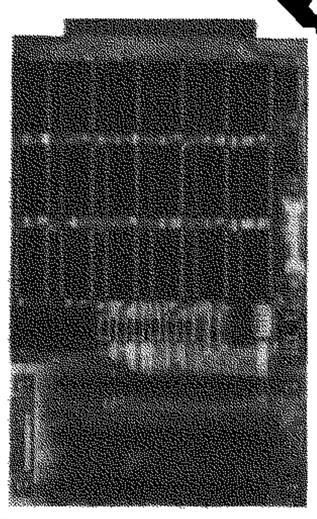
Evade Self Destruct Code With Write Protect Switch

All Boards Factory Tested

Complete Instructions And Illustrations Included.

Fuse Protected

Reset Switch



Pictured Above –
Action Board with all options

24K BOARD FEATURES:

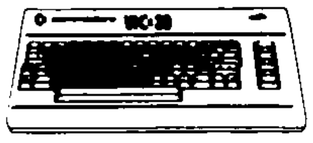
- *Adds 24K Memory (29K with VIC* 5K).
- *Upgrade Board to 35K by adding IC's and switches.
- *Memory switchable in 8K sections. (No need to remove memory board to run your other programs).
- *3 expansion slots with switches (for game or extra utility cartridges).
- *Reset button allows restarting computer without turning power off.
- *.5 amp fuse protected.
- *Switch relocates expansion cartridges in memory so that it can be saved on tape as a backup for your valuable programs. (The unexpanded VIC will not allow cartridges to be saved on tape).
- *Write protect switches allow programs stored in RAM at ROM location or entire Memory to be protected against accidental write.
- *Switch allows memory to be moved between RAM and ROM location. (Useful for developing your own games and saving on tape).
- *Gold plated card edge connector.
- *No other memory expansion needed.
- *Easily plugs into your VIC, no modifications necessary. Saves wear on your VIC 20 since board never needs to be removed or power turned off and on to run other tapes or cartridges.
- **Optional 35K memory (40K with VIC* 5K).

24K memory, 3 expansion slots, switches assembled and tested **$109.95**

Same as above with sockets that allow you to later add your own memory chips to bring memory from 24K to 35K **$124.95**

Full 35K memory, 3 expansion slots, 3K expander mode, eprom socket (switch selectable between BLK 3 & BLK 5) and all switches assembled and tested (eprom not included) **$149.95**

Bare 35K board with complete instruction and parts list **$39.95**



VIC 20* is a registered trademark of Commodore.

COD Orders: 816–444–4651

MIDWEST PERIPHERAL INDUSTRIES

MPI

Add $5 for shipping and handling
Mo. residents add 5½% sales tax
Send check or money order to

MPI
Box 8123-B
Kansas City, MO 64112

Personal checks – Allow 3 weeks to clear.

NEW

VIC-20

GOSUB

C64

INTERNATIONAL INCORPORATED

The Flexikey System

Retail
$69.95

Features:

19 Keys, each of which may have 3 seperate definitions!

Complete documentation including program listings!

Works on the VIC20 (Expanded) and C-64 computers!

Compatible with most existing software!

Great for use with business programs and electronic spread sheets!

Ideal for machine language programmer!

VISA & MASTERCARD WELCOME

Prices subject to change

Dealer Inquires Invited - **(316) 265-9858**
GOSUB International - 501 E. Pawnee - Suite 430
Wichita, Kansas 67211

*C-64 and VIC 20 are registered trademarks of Commodore International.

COMMODORE USERS

Join the largest, active Commodore users group in North America and get—

— Access to club library of over 3000 free programs.

— Informative club newsletter.

— The latest information about the PET, CBM, VIC, Super-PET and Commodore-64.

Send $20.00 ($30.00 overseas) for Associate Membership to:

Toronto Pet Users Group
Department "M"
381 Lawrence Avenue West
Toronto, Ontario, Canada M5M 1B9

This Space Could Be Transacting For You!

Advertising Index

Software

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Academy Software	VIC20/C64 Software					**73**		
Advantage Computer Acc.	VIC 20 games				75			
Beacon Software	Business Packages				74			
Boston Educational Computing	Educational Software					**64**		
Computer Marketing\Canadian Micro	Calc Result (spreadsheet prog.)	Handic Software ab		IBC	IBC	**IBC**		
Data 20 Corp.	Word Manager					**67**		
Eastern House	MAE Assembler					**68**		
Execom Corp.	40/80 Screen Select					**68**		
Input Systems Inc.	Typro (wordprocessor)					**68**		
Magreeable Software	Stock Helper					**66**		
Micro Applications	Master (programming aid)		IBC					
Midwest Micro Inc.	VIC20/C64 Graphics Util					**69**		
Pacific Coast Software	C64: Utility, Business, Games			62				
Performance Micro Products	C64 Forth					**64**		
Precision Software	Superscript (wordprocessor)		61					
Pro-Line Software	PAL 64 (assembler)				79	**73**		
	POWER 64 (programming aid)				76	**69**		
	MailPro					**74**		
	general				76	**66**		
Software International	VIC20/C64 Software					**65**		
Southern Solutions	Business packages				78	**71**		
William Robbins Software	VIC/64/PET Software					**74**		
Wycor Business Systems	Provincial Payroll			64				

Hardware

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Computer Workshops \Computer Marketing	Z-RAM (CP/M board)	Madison		63				
Connecticut microComputer	PET/CBM Interface adapters			62	79			
	Analog/Digital I/O					**64**		
Eastern House	Trap 65					**68**		
	VIC Rabbit					**68**		
	Eprom Programmer					**68**		
	Communications Bd					**68**		
George M. Drake & Associates	Colour Monitors	Amdek				**BC**		
Gosub International	Flexikey					**75**		
Micro Ware	Tape Interface					**70**		
	VIC20 RAM Expand					**70**		
Micro World Electronix	VIC20/C64 Printer Interface					**65**		
Midwest Micro Inc.	Smart ASCII Plus					**69**		
Midwest Peripherals	VIC20 Expander					**74**		
Precision Technology	VIC20/C64 Expander Boards			61	79	**73**		
Richvale Telecommunications	C64 Link (IEEE adapter)			IFC	IFC	**IFC**		

Accessories

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Advantage Computer Acc.	Joysticks	Wico			77			
Computer Workshops	Apr83 products list				76			
Consultors Int'l	Stock Market With Your PC (book)		64					
"	A To Z Book Of Games		64					
"	Dynamics Of Money Mgmt (book)		64					
"	Invment. Analysis w/Your Micro (book)		64					
Cursor	Tape Magazine					**69**		
Leading Edge Inc.	Elephant Diskettes		BC	BC	BC			
Midnight Software Gazette	Subscriber Info					**72**		
Toronto PET Users Group	Membership info			61		**75**		

A More Powerful Planning
And Forecasting Tool Than Any Other On The Market . . .

Calc Result

A three-dimensional spread sheet
with multiple pages of 63 x 254 cells
which utilizes only the memory
in cells that are active

Produces Graphics (Histograms) on screen
and printer

Gives unlimited possibilities in each cell with
IF-THEN-ELSE with AND, OR and NOT-ELSE

Available now for 8032, 8096 and Commodore 64.

FOR FURTHER INFORMATION CONTACT YOUR NEAREST COMMODORE DEALER

OR

CANADIAN MICRO DISTRIBUTORS
500 Steeles Avenue
Milton, Ontario, Canada L9T 3P7
(416) 876-4741

CALC RESULT *is a trademark of Handic Software ab*

COMPATIBLE COLOR-I . . .

NEW 2 YEAR WARRANTY!
On all monitor electronics . . . 3 yrs. on all CRT's
(See details at dealer)

The popular choice for popular computers . . . at a popular price.

The Color-I Monitor is designed to perform superbly with your Apple II, Atari or VIC Commodore personal computer and others. Highly styled cabinet. It accepts a composite video signal to produce vivid, richly colored graphic and sharp text displays. Very reasonably priced, the Color-I is a giant step above home TV sets and other monitors.

Just write, or call to receive complete specifications on the Amdek Color-I Monitor.

- Quality 260(H) x 300(V) line resolution.
- Built-in speaker and audio amplifier.
- Front mounted controls for easy adjustment.
- Interface cables available for Atari and VIC Commodore computers.
- FCC/UL approved.

2201 Lively Blvd. • Elk Grove Village, IL 60007
(312) 364-1180 TLX: 25-4786
REGIONAL OFFICES: Calif. (714) 662-3949 • Texas (817) 498-2334

AMDEK CORP.

Amdek . . . your guide to innovative computing!